Configuration Manual

MSc Research Project
Cybersecurity

Ronan Shaw
Student ID: 22129618

School of Computing
National College of Ireland

Supervisor: Noel Cosgrave

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Ronan Shaw
Student ID:	22129618
Programme:	MSCCYB1 **Year:** 2022
Module:	Research Project
Supervisor:	Noel Cosgrave
Submission Due Date:	14th August 2023
Project Title:	Identification of malicious domains based on temporal features of X.509 certificates and registrar records.
Word Count:	2115 excl. cover, refs, appendices. **Page Count:** 15 excl. appendices.

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Ronan Shaw

Date: 11th August 2023

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Ronan Shaw
Student ID: 22129618

Contents

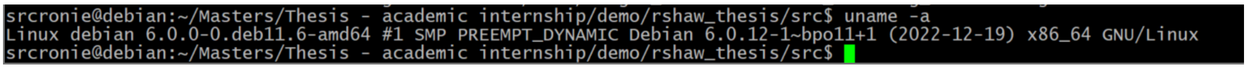
Hardware requirements.

- Processor: AMD Ryzen 7 2700 Eight-Core Processor. (3.2GHz base clock) (AMD64)
- Memory 32GB.
- Storage: 1TB SSD (NVME), 3TB HDD.
- Networking: 250Mbps cable broadband.

Operating system.

- Debian Linux 11.7 (*bullseye*)

```
$ uname -a
Linux debian 6.0.0-0.deb11.6-amd64 #1 SMP PREEMPT_DYNAMIC Debian
6.0.12-1~bpo11+1 (2022-12-19) x86_64 GNU/Linux
```



```
srcronie@debian:~/Masters/Thesis - academic internship/demo/rshaw_thesis/src$ uname -a
Linux debian 6.0.0-0.deb11.6-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.0.12-1~bpo11+1 (2022-12-19) x86_64 GNU/Linux
srcronie@debian:~/Masters/Thesis - academic internship/demo/rshaw_thesis/src$
```

Figure 1 - uname

Release notice: https://www.debian.org/News/2023/20230429
ISO images: https://www.debian.org/releases/bullseye/debian-installer/

A full list of OS packages install is provided in **pkglist.txt** the source repository.

If required in the case of dependency issues, these can be installed as follows, note these will include software which is not used by the project and it is recommended that it is installed in a virtual machine so as to not impact your system:

```
$ xargs sudo apt-get -y install < ./src/pkglist.txt
```

Source code.

Source code and sample data is available in the uploaded artefact zip file.
The source code in the artefact is the final version.

A complete source code history is available at https://github.com/roniencirl/thesis.git

Setup a new directory and extract the archive:

```
$ mkdir demo
```

Within this directory we will extract the artefact: rshaw_thesis_code_artefact.zip

Verify the hash:

```
$ sha256sum rshaw_thesis_code_artefact.zip
43224c538d7b95deefa9c89a19a319fa27cba84ebfb80fbc3c497e4061fbc0f5
rshaw_thesis_code_artefact.zip
```


```
srcronie@debian:~/Masters/Thesis - academic internship/demo$ sha256sum rshaw_thesis_code_artefact.zip
43224c538d7b95deefa9c89a19a319fa27cba84ebfb80fbc3c497e4061fbc0f5  rshaw_thesis_code_artefact.zip
srcronie@debian:~/Masters/Thesis - academic internship/demo$
```

Figure 2 - artefact sha256 hash

Unzip the artefact:

```
$ unzip rshaw_thesis_code_artefact.zip
```

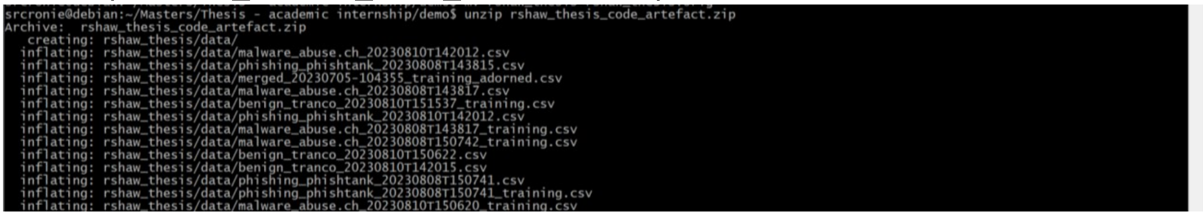

```
srcronie@debian:~/Masters/Thesis - academic internship/demo$ unzip rshaw_thesis_code_artefact.zip
Archive:  rshaw_thesis_code_artefact.zip
   creating: rshaw_thesis/data/
  inflating: rshaw_thesis/data/malware_abuse.ch_20230810T142012.csv
  inflating: rshaw_thesis/data/phishing_phishtank_20230808T143815.csv
  inflating: rshaw_thesis/data/merged_20230705-104355_training_adorned.csv
  inflating: rshaw_thesis/data/malware_abuse.ch_20230808T143817.csv
  inflating: rshaw_thesis/data/benign_tranco_20230810T151537_training.csv
  inflating: rshaw_thesis/data/phishing_phishtank_20230810T142012.csv
  inflating: rshaw_thesis/data/malware_abuse.ch_20230808T143817_training.csv
  inflating: rshaw_thesis/data/malware_abuse.ch_20230808T150742_training.csv
  inflating: rshaw_thesis/data/benign_tranco_20230810T150622.csv
  inflating: rshaw_thesis/data/benign_tranco_20230810T142015.csv
  inflating: rshaw_thesis/data/phishing_phishtank_20230808T150741.csv
  inflating: rshaw_thesis/data/phishing_phishtank_20230808T150741_training.csv
  inflating: rshaw_thesis/data/malware_abuse.ch_20230810T150620_training.csv
```

Figure 3 - unzip output sample

Export the project path for future commands:

```
$ cd rshaw_thesis
$ export PROJ_PATH=$(pwd)
```

Filesystem layout.

The ZIP artefact includes the following directories:

```
.
├── data
├── datasets
│   ├── abuse.ch
│   ├── ct_logs
│   │   ├── certs
│   │   │   └── certificates
│   │   │       └── https:__ct.cloudflare.com_logs_nimbus2023
│   │   ├── done
│   │   └── tmp
│   │       └── certificates
│   │           └── https:__ct.cloudflare.com_logs_nimbus2023
│   ├── phishtank
│   ├── public_suffix_list
│   └── tranco_1m
├── jupyter_rendered
└── src
    └── docker
```

- data: contain the interim CSV files produced and consumed by
  - the domain collation script: `setup-training-data.py`
  - the feature adornment script: `prepare-training-data-parallel.py`
  - the jupyter notebooks for the Logistic Regression and Random Forest models.
- datasets:
  - abuse.ch, phishtank, trance_1m : the benign and malicious domain datasets
  - ct_logs: a small, pre-downloaded sample of the certificate transparency logs which Axeman utility would download and the `ctlog-to-mongo.py` script will process.
- jupyter_rendered: HTML rendered version of the Jupyter Notebooks.
- src:
  - `Pipfile` & `Pipfile.lock` for the pipenv dependencies.
  - The python scripts and their `utils.py` library.
  - Config file for the dataset csv layout: `datasets_config.py`
  - Script to load CTLog attributes into DB: `ctlog-to-mongo.py`
  - The domain collation script: `setup-training-data.py`
  - The feature adornment script: `prepare-training-data-parallel.py`
  - Pipeline to render all jupyter notebooks: `render_all_notebooks.py`
  - Docker compose for mongodb and memcache: `docker/docker-compose.yml`
  - Debian 11.7 packages installed on original machine: `pkglist.txt`
  - Jupyter notebooks, listed in Table 1.

Table 1 - Jupyter Notebooks

Logistic Regression Model Notebooks	Random Forest Model Notebooks
`sm model 3 lr - 40k features A.ipynb` `sm model 3 lr - 40k features baseline.ipynb`	`sklearn model rf 4 - 40k features A.ipynb` `sklearn model rf 4 - 40k features baseline.ipynb`

sm model 3 lr - 40k features B.ipynb	sklearn model rf 4 - 40k features B.ipynb
sm model 3 lr - 40k features C.ipynb	sklearn model rf 4 - 40k features C.ipynb
sm model 3 lr - 40k features D.ipynb	sklearn model rf 4 - 40k features D.ipynb
sm model 3 lr - 40k features E.ipynb	sklearn model rf 4 - 40k features E.ipynb
sm model 3 lr - 40k features F.ipynb	sklearn model rf 4 - 40k features F.ipynb
sm model 3 lr - 40k features G.ipynb	sklearn model rf 4 - 40k features G.ipynb
sm model 3 lr - 40k features H.ipynb	sklearn model rf 4 - 40k features H.ipynb

Docker.

Docker is required for the MongoDB and Memcached container instances. A docker-compose.yml file is included to initalise them with the correct network and credentials configuration.

Docker install.

Docker is installed (Docker version 24.0.2, build cb74dfc).
The /var/lib/docker directory symbolically linked to the 1TB NVME storage detailed above.

Note: *This is optional if the system had sufficient fast storage already.*

This can be achieved by:
Stopping the docker service:
`$ sudo systemctl stop docker:`
Moving files from /var/lib/docker to new directory
`$ sudo rsync -a --sparse --progress /var/lib/docker /nvme/docker`
Creating a symlink:
`$ rm –rf /var/lib/docker; ln –s /nvme/docker /var/lib/docker`
Start docker service:
`$ sudo systemctl start docker`

For example:

```
ronie@debian:~$ ls -la /var/lib/docker
lrwxrwxrwx 1 root root 16 Jun 11 15:26 /var/lib/docker -> /nvmepci/docker/
```

Docker containers install.

A Docker MongoDB container is used to store the relevant fields parsed from the downloaded certificate transparency logs and requires high speed storage to be performant (Harazdovskiy, 2022; *Performance Best Practices: Hardware and OS Configuration | MongoDB Blog*, no date).

The artefact sha256 hashes are included here to ensure the correct version of container images are used.

```
$ docker image pull
mongo@sha256:fb0d2e1151b2de653e1d793515a90c09d0aeaa7807b1184be6f5a69
81432cc92
```

A Docker Memcached container is used to cache the results of Whois in memory as well as crt.sh lookups in the case of failure to find the certificate fields in MongoDB.

```
$ docker image pull
memcached@sha256:0451628d4eced8a4b7f74946bae89e4bb401cb7dc2d7547b408
7c3e40159dc25
```

To launch both containers use the `${PROJ_PATH}/src/docker/docker-compose.yml` configuration file which is preconfigured to forward the required ports for both and with username/password set for MongoDB.

```
$ docker compose -f ${PROJ_PATH}/src/docker/docker-compose.yml

$ docker start memcache mongo
```

```
srcronie@debian:~/Masters/Thesis - academic internship/demo/rshaw_thesis/src$ docker start mongo memcache
mongo
memcache
srcronie@debian:~/Masters/Thesis - academic internship/demo/rshaw_thesis/src$ docker ps
CONTAINER ID   IMAGE       COMMAND                  CREATED        STATUS         PORTS                                           NAMES
1c2ecdee5d28   mongo       "docker-entrypoint.s…"   2 months ago   Up 5 seconds   0.0.0.0:27017->27017/tcp, :::27017->27017/tcp   mongo
f043320f3cab   memcached   "docker-entrypoint.s…"   2 months ago   Up 2 seconds   0.0.0.0:11211->11211/tcp, :::11211->11211/tcp   memcache
srcronie@debian:~/Masters/Thesis - academic internship/demo/rshaw_thesis/src$
```

Figure 4 - example of running docker containers

Python environment.

Tooling and associated configuration is written in Python version 3.9.2. In order to ensure portability a Python virtualenv is created using pipenv.

Pip version: 23.2.1
Setuptools version: 68.0.0
Pipenv version: 2023.7.23
Virtualenv version: 20.24.2

A full list of the required Python modules is in the ./src/Pipfile file and will be automatically installed in the virtualenv when created by pipenv.

1. Install latest Pip (or update to latest)
   ```
   $ python3 -m pip install -U pip
   ```

2. Install latest versions
   Pipenv: https://pipenv.pypa.io/en/latest/
   Virtualenv: https://virtualenv.pypa.io/en/latest/
   Setuptools: https://pypi.org/project/setuptools/
   ```
   $ python3 -m pip install –U --user pipenv pip setuptools virtualenv
   ```

3. Change to the ./src/ directory enter pipenv shell
   ```
   $ python3 –m pipenv shell --python 3.9.2
   ```

4. Install the pipenv dependencies found in the Pipfile.
   ```
   $ pipenv install --verbose
   ```
   This may take some time (20-30 mins), in particular due to the axeman utility being downloaded from github.
   While it running the following will be shown:

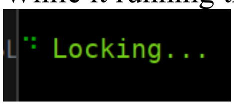
   

   Figure 5 - "snake" Locking animation for pipenv install

   If it fails try clearing the lock and running again.
   ```
   $ pipenv lock --clear
   ```
   cache and running again.
   ```
   $ pipenv --rm
   $ pipenv install --clear –verbose
   ```

An example of a successful pipenv install output is given in Figure 6.

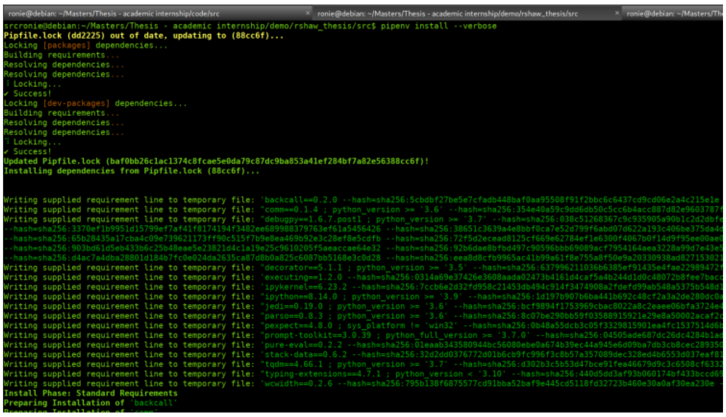


Figure 6- Successful Pipenv Install output

Downloading datasets.

Note: *Downloading the datasets is not required to run the models using the pre-processed data, it is included here for completeness.*

Skip to here to use the supplied pre-processed data with the models.

Due to the size of the Certificate Transparency log this dataset is stored separately (on a 3TB HDD for this report) and only a small sample is included in the zip artefact.
Instructions to download the entire CT Log are included in **here**.

Abuse.ch

To download a newer version of the Abuse.ch dataset.
From the root of the repo download and extract the zip file:

```
cd ./dataset/abuse.ch
curl https://urlhaus.abuse.ch/downloads/csv/ -o abuse_malware.zip
unzip  abuse_malware.zip
ls  csv.txt
```

Phishtank

Direct programmatic download or API access to this data source is not currently available at the time of writing.
Manually downloaded http://data.phishtank.com/data/online-valid.csv from https://phishtank.org/developer_info.php

Extract and save in ${PROJ_PATH}/datasets/phishtank/verified_online.csv

Tranco

The Tranco Umbrella list was used, from https://umbrella-static.s3-us-west-1.amazonaws.com/top-1m.csv.zip.

```
$ cd ${PROJ_PATH}/datasets/tranco
```

```
$ curl https://umbrella-static.s3-us-west-1.amazonaws.com/top-1m.csv.zip -o top-1m.csv.zip
```

This was extracted as follows:

```
$ unzip top-1m.csv.zip
$ mv top-1m.csv "top-1m umbrella.csv"
```

Public Suffix List

This is used when parsing domains to establish if we have reached the Top Level Domain or not.

```
cd ${PROJ_PATH}/datasets/public_suffix_list/
curl
https://raw.githubusercontent.com/publicsuffix/list/master/public_suffix_list.dat -o public_suffix_list.dat
```

Certificate Transaction Log Dataset.

Certificate transparency logs are downloaded using a fork of the Axeman utility.

The repository for the project contains a pipenv Python virtualenv configuration file, to ensure that there are no missing modules or dependencies all further actions should be carried out within the pipenv shell environment as detailed in here.

Note: Retrieval of the full CT Log Axeman is not required to load the provided sample CT Log data from the zip artefact into the database, but is required if the running the full data retrieval and dataset feature adornment process.

Install Axeman from Github

Optional – Axeman is already included in the pipenv Pipfile dependencies.

```
$ cd ${PROJ_PATH}/datasets/ct_logs/
$ python3 –m pipenv shell
$ python3 –m pip install git+https://github.com/roniencirl/Axeman.git
```

Prepare directories

```
$ mkdir ${PROJ_PATH}/datasets/ct_logs
```

Run Axeman to retrieve the Certificate Transparency Logs.

This will potentially take a number of days to complete.

```
$ cd ${PROJ_PATH}/datasets/ct_logs/
$ python3 –m pipenv shell
$ cd ../datasets/ct_logs/
$ axeman -o ./certs/ -t ./tmp/ -u
https://ct.cloudflare.com/logs/nimbus2023
```

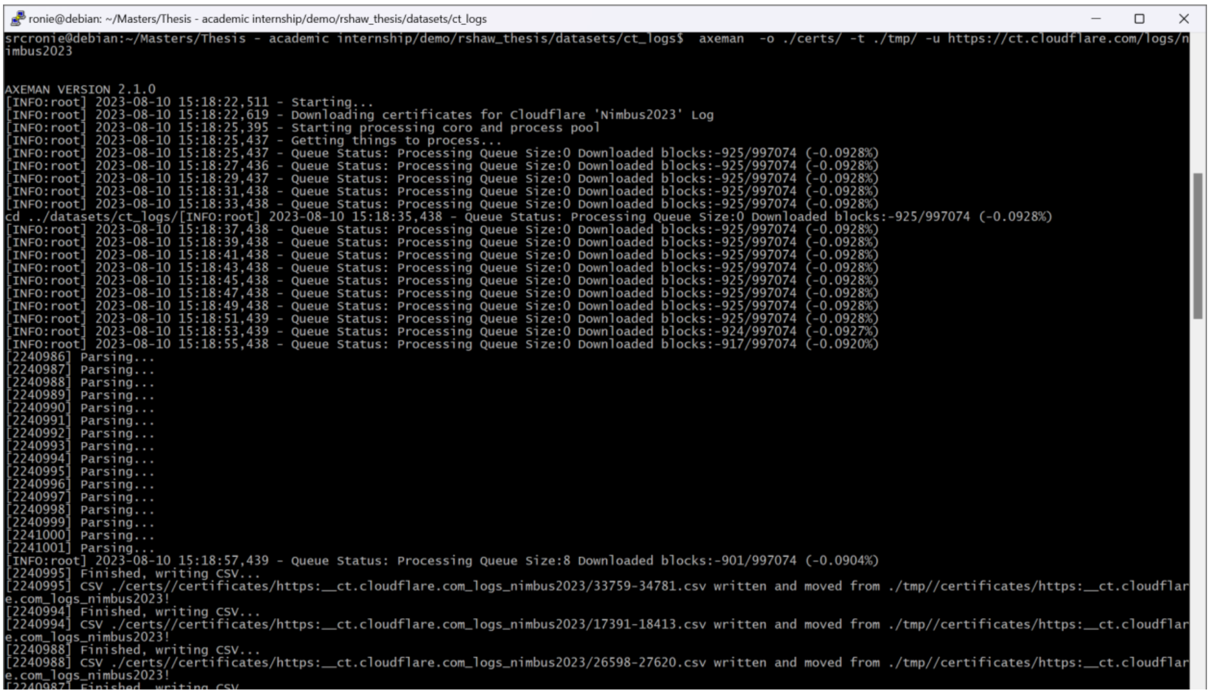


Figure 7 – Successful Axeman execution

Figure 7 shows a successful Axeman execution in progress.
If at any time the Axeman download fails, which can occur, the index of the last file downloaded can be used to continue the download process from that point:

```
$ ls
${PROJ_PATH}/datasets/ct_logs/certs/certificates/https\:__ct.cloudflare.com_logs_nimbus2023/
-rw-r--r-- 1 ronie ronie      44 Jul  4 05:02 489367395-489368417.csv

$ axeman -z 489367395 -o ./certs/ -t ./tmp/ -u
https://ct.cloudflare.com/logs/nimbus2023
```

Extract relevant certificate log attributes into database.

Each domains earliest certificate valid from date and latest certificate valid to date are extracted from the csv files into the mongoDB database. The zip artefact contains some sample CT Log files to demonstrate this process.

As processed files are moved out of the original directory to `{PROJ_PATH}/datasets/ct_logs/done` this can be executed in parallel with the Axeman Certificate Transaction Log download process.

Run the script to extract the attributes to mongoDB, Figure 8:

```
$ cd {PROJ_PATH}/src/
$ python3 –m pipenv shell
$ python ./ctlog-to-mongo.py
```

Figure 8 - CTLog to MongoDB execution

Generate a list of domains.

The Abuse.ch, Phishtank and Tranco datasets are in different formats, we extract the relevant fields for each and setup a primary dataset, for our list of training and test domains, of the size required, Figure 9.

```
$ cd {PROJ_PATH}/src
$ python3 -m pipenv shell
$ python setup-training-data.py
```

```
srcronie@debian:~/Masters/Thesis - academic internship/demo/rshaw_thesis/src$ python ./setup-training-data.py
./phishtank/verified_online.csv
(56517, 8)
Wrote 56517 rows to ../data/phishing_phishtank_20230810T150619.csv
./abuse.ch/csv.txt
(16370, 9)
Wrote 16370 rows to ../data/malware_abuse.ch_20230810T150620.csv
./tranco_1m/top-1m umbrella.csv
(999999, 2)
Wrote 100000 rows to ../data/benign_tranco_20230810T150622.csv
```

Figure 9 - Generate list of domains for the training and test data

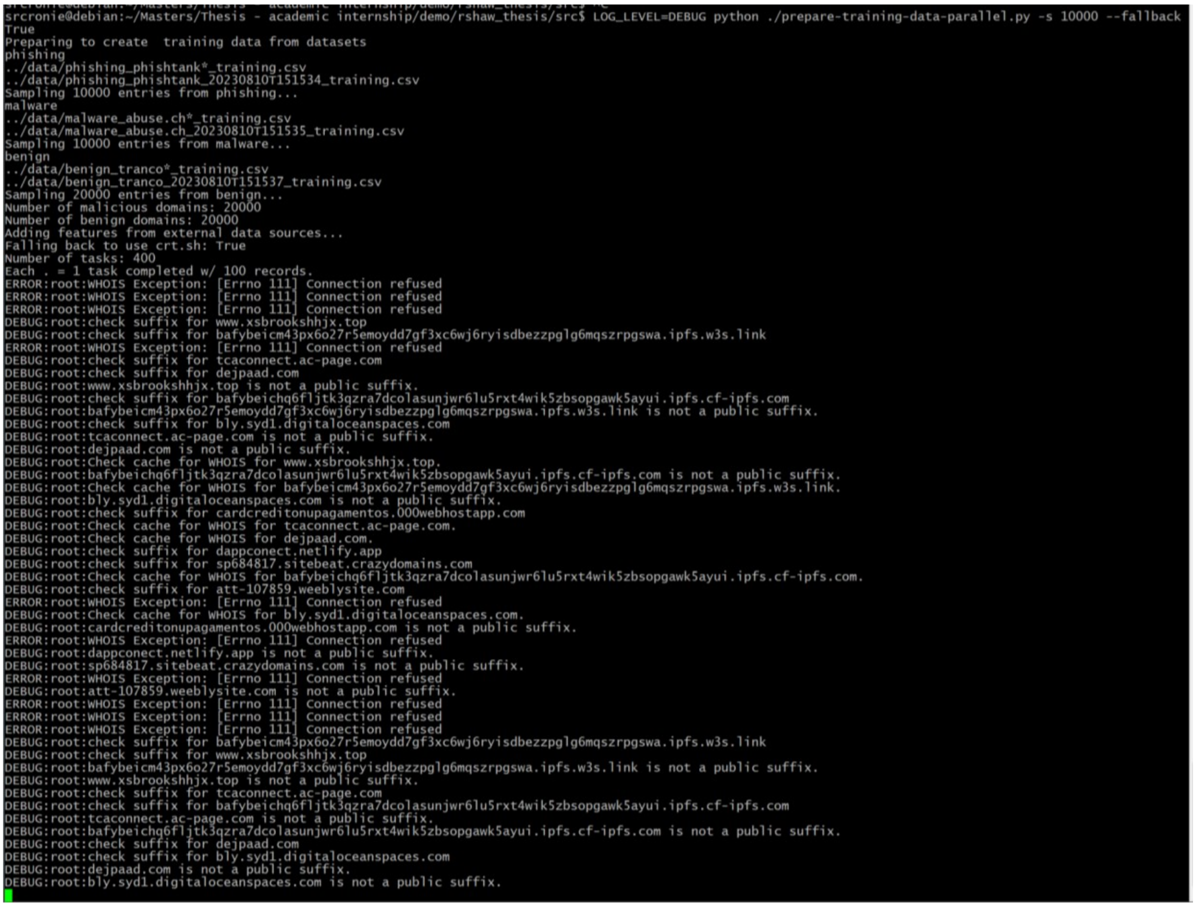
These csv files are used by the subsequent steps.

Lookup and save the temporal features for each domain.

This step adds the temporal features by lookup of the domain from Whois and the CT Log CN, SAN and Validity attributes in the MongoDB, falling back to crt.sh if required – Figure 10.

The lookups are performed in parallel for groups of domains and is processor, memory, and network intensive.
Smaller samples of domains can be used to add temporal features to the domain dataset using the -s switch.

```
$ cd {PROJ_PATH}/src
$ python3 –m pipenv shell
$ python3 ./prepare-training-data-parallel.py -s 10000 -b True
```



```
srcronie@debian:~/Masters/Thesis - academic internship/demo/rshaw_thesis/src$ LOG_LEVEL=DEBUG python ./prepare-training-data-parallel.py -s 10000 --fallback True
Preparing to create  training data from datasets
phishing
../data/phishing_phishtank*_training.csv
../data/phishing_phishtank_20230810T151534_training.csv
Sampling 10000 entries from phishing...
malware
../data/malware_abuse.ch*_training.csv
../data/malware_abuse.ch_20230810T151535_training.csv
Sampling 10000 entries from malware...
benign
../data/benign_tranco*_training.csv
../data/benign_tranco_20230810T151537_training.csv
Sampling 20000 entries from benign...
Number of malicious domains: 20000
Number of benign domains: 20000
Adding features from external data sources...
Falling back to use crt.sh: True
Number of tasks: 400
Each . = 1 task completed w/ 100 records.
ERROR:root:WHOIS Exception: [Errno 111] Connection refused
ERROR:root:WHOIS Exception: [Errno 111] Connection refused
ERROR:root:WHOIS Exception: [Errno 111] Connection refused
DEBUG:root:check suffix for www.xsbrookshhjx.top
DEBUG:root:check suffix for bafybeicm43px6o27r5emoydd7gf3xc6wj6ryisdbezzpglg6mqszrpgswa.ipfs.w3s.link
ERROR:root:WHOIS Exception: [Errno 111] Connection refused
DEBUG:root:check suffix for tcaconnect.ac-page.com
DEBUG:root:check suffix for dejpaad.com
DEBUG:root:www.xsbrookshhjx.top is not a public suffix.
DEBUG:root:check suffix for bafybeichq6fljtk3qzra7dcolasunjwr6lu5rxt4wik5zbsopgawk5ayui.ipfs.cf-ipfs.com
DEBUG:root:bafybeicm43px6o27r5emoydd7gf3xc6wj6ryisdbezzpglg6mqszrpgswa.ipfs.w3s.link is not a public suffix.
DEBUG:root:check suffix for bly.syd1.digitaloceanspaces.com
DEBUG:root:tcaconnect.ac-page.com is not a public suffix.
DEBUG:root:dejpaad.com is not a public suffix.
DEBUG:root:Check cache for WHOIS for www.xsbrookshhjx.top.
DEBUG:root:bafybeichq6fljtk3qzra7dcolasunjwr6lu5rxt4wik5zbsopgawk5ayui.ipfs.cf-ipfs.com is not a public suffix.
DEBUG:root:Check cache for WHOIS for bafybeicm43px6o27r5emoydd7gf3xc6wj6ryisdbezzpglg6mqszrpgswa.ipfs.w3s.link.
DEBUG:root:bly.syd1.digitaloceanspaces.com is not a public suffix.
DEBUG:root:check suffix for cardcreditonupagamentos.000webhostapp.com
DEBUG:root:Check cache for WHOIS for tcaconnect.ac-page.com.
DEBUG:root:Check cache for WHOIS for dejpaad.com.
DEBUG:root:check suffix for dappconect.netlify.app
DEBUG:root:check suffix for sp684817.sitebeat.crazydomains.com
DEBUG:root:Check cache for WHOIS for bafybeichq6fljtk3qzra7dcolasunjwr6lu5rxt4wik5zbsopgawk5ayui.ipfs.cf-ipfs.com.
DEBUG:root:check suffix for att-107859.weeblysite.com
ERROR:root:WHOIS Exception: [Errno 111] Connection refused
DEBUG:root:Check cache for WHOIS for bly.syd1.digitaloceanspaces.com.
DEBUG:root:cardcreditonupagamentos.000webhostapp.com is not a public suffix.
ERROR:root:WHOIS Exception: [Errno 111] Connection refused
DEBUG:root:dappconect.netlify.app is not a public suffix.
DEBUG:root:sp684817.sitebeat.crazydomains.com is not a public suffix.
ERROR:root:WHOIS Exception: [Errno 111] Connection refused
DEBUG:root:att-107859.weeblysite.com is not a public suffix.
ERROR:root:WHOIS Exception: [Errno 111] Connection refused
ERROR:root:WHOIS Exception: [Errno 111] Connection refused
ERROR:root:WHOIS Exception: [Errno 111] Connection refused
DEBUG:root:check suffix for bafybeicm43px6o27r5emoydd7gf3xc6wj6ryisdbezzpglg6mqszrpgswa.ipfs.w3s.link
DEBUG:root:check suffix for www.xsbrookshhjx.top
DEBUG:root:bafybeicm43px6o27r5emoydd7gf3xc6wj6ryisdbezzpglg6mqszrpgswa.ipfs.w3s.link is not a public suffix.
DEBUG:root:www.xsbrookshhjx.top is not a public suffix.
DEBUG:root:check suffix for tcaconnect.ac-page.com
DEBUG:root:check suffix for bafybeichq6fljtk3qzra7dcolasunjwr6lu5rxt4wik5zbsopgawk5ayui.ipfs.cf-ipfs.com
DEBUG:root:tcaconnect.ac-page.com is not a public suffix.
DEBUG:root:bafybeichq6fljtk3qzra7dcolasunjwr6lu5rxt4wik5zbsopgawk5ayui.ipfs.cf-ipfs.com is not a public suffix.
DEBUG:root:check suffix for dejpaad.com
DEBUG:root:check suffix for bly.syd1.digitaloceanspaces.com
DEBUG:root:dejpaad.com is not a public suffix.
DEBUG:root:bly.syd1.digitaloceanspaces.com is not a public suffix.
```

Figure 10 - Retrieve features for the domains

This will produce csv files in ./data/ with all the required fields for the training and testing of the models – Figure 11.

```
$  ls ${PROJ_PATH}/data/merged_*_training_adorned-engineered.csv
```

```
srcronie@debian:~/Masters/Thesis - academic internship/demo/rshaw_thesis/src$ ls -lrt ../data/merged_*_training_adorned-engineered.csv
-r--r--r-- 1 ronie ronie 5073004 Aug 10 10:42 ../data/merged_20230705-104357_training_adorned-engineered.csv
srcronie@debian:~/Masters/Thesis - academic internship/demo/rshaw_thesis/src$
```

Figure 11 - example of prepared dataset provided

Training & testing the models.

A set of Jupyter notebooks are provided along with a script to pipeline execution and rendering of them using the prepared merged_*_training_adorned-engineered.csv file.

Setup Jupyter

Optional – Jupyter modules are included in the pipenv Pipfile dependencies.

```
$ cd {PROJ_PATH}/src
$ python3 -m pipenv shell
$ python3 -m pip install jupyter-server jupyter-client
```

Version installed:

```
$ pip list | grep jupyter
jupyter_client            7.4.9
jupyter_core              5.3.1
jupyter-events            0.6.3
jupyter_server            2.7.0
jupyter_server_terminals  0.4.4
jupyterlab-pygments       0.2.2

$ jupyter --version
Jupyter versions:
IPython          : 8.14.0
ipykernel        : 6.23.2
ipywidgets       : not installed
jupyter_client   : 7.4.9
jupyter_core     : 5.3.1
jupyter_server   : 2.7.0
jupyterlab       : not installed
nbclient         : 0.8.0
nbconvert        : 7.6.0
nbformat         : 5.9.0
notebook         : 6.5.4
qtconsole        : not installed
traitlets        : 5.9.0
```

Running the models.

Two sets of jupyter files for performing the training and testing of the model and producing the statistical results.

- sm model 3 lr - 40k features *.ipynb : Statsmodel Logistic Regression
- sklearn model 3 rf - 40k features A.ipynb: SciKit Learn Logistic Regression

If the preconfigured default prepared dataset in the Jupyter notebooks
${PROJ_PATH}/data/merged_20230705-104357_training_adorned-engineered.csv
is not used then modify the training data file variable in the jupyter notebooks to point to a specific file or use a wildcard to select the most recent matching file.

```
    ]

path = "../data/"
filename = None

epoch = datetime(1970,1,1)
# open the training data file, from ../data/ for the most recent merged training data file saved by prepare-training-data-parallel.py
full_path = path + "merged_20230705-104357_training_adorned-engineered.csv"

```

Figure 12 - Example showing where to change the dataset chosen in the *.ipynb files.

All models can be executed with the render pipeline script provided, or a subset using shell wildcards for example:

```
$ cd {PROJ_PATH}/src
$ python3 -m pipenv shell
$ python ./render_all_notebooks.py --execute true --notebook_path
"./*features*.ipynb"
```

```
PROBLEMS 323   OUTPUT   DEBUG CONSOLE   TERMINAL   JUPYTER   GITLENS

(src-yWZ-XGoe) ronie@debian:~/Masters/Thesis - academic internship/code/src$ python ./render_all_notebooks.py --execute true --notebook_path "./*lr*features*.ipynb"
[NbConvertApp] Converting notebook ./sm model 3 lr - 40k features baseline.ipynb to html
[NbConvertApp] Writing 1048583 bytes to sm model 3 lr - 40k features baseline.html
rendered ./sm model 3 lr - 40k features baseline.ipynb->../jupyter_rendered/./sm model 3 lr - 40k features baseline_20230810141730.html
[NbConvertApp] Converting notebook ./sm model 3 lr - 40k features A.ipynb to html
[NbConvertApp] Writing 1049607 bytes to sm model 3 lr - 40k features A.html
rendered ./sm model 3 lr - 40k features A.ipynb->../jupyter_rendered/./sm model 3 lr - 40k features A_20230810141730.html
[NbConvertApp] Converting notebook ./sm model 3 lr - 40k features B.ipynb to html
[NbConvertApp] Writing 1050522 bytes to sm model 3 lr - 40k features B.html
rendered ./sm model 3 lr - 40k features B.ipynb->../jupyter_rendered/./sm model 3 lr - 40k features B_20230810141730.html
[NbConvertApp] Converting notebook ./sm model 3 lr - 40k features E.ipynb to html
[NbConvertApp] Writing 1051476 bytes to sm model 3 lr - 40k features E.html
rendered ./sm model 3 lr - 40k features E.ipynb->../jupyter_rendered/./sm model 3 lr - 40k features E_20230810141730.html
[NbConvertApp] Converting notebook ./sm model 3 lr - 40k features D.ipynb to html
[NbConvertApp] Writing 1049311 bytes to sm model 3 lr - 40k features D.html
rendered ./sm model 3 lr - 40k features D.ipynb->../jupyter_rendered/./sm model 3 lr - 40k features D_20230810141730.html
[NbConvertApp] Converting notebook ./sm model 3 lr - 40k features C.ipynb to html
[NbConvertApp] Writing 1050682 bytes to sm model 3 lr - 40k features C.html
rendered ./sm model 3 lr - 40k features C.ipynb->../jupyter_rendered/./sm model 3 lr - 40k features C_20230810141730.html
[NbConvertApp] Converting notebook ./sm model 3 lr - 40k features H.ipynb to html
[NbConvertApp] Writing 1076415 bytes to sm model 3 lr - 40k features H.html
rendered ./sm model 3 lr - 40k features H.ipynb->../jupyter_rendered/./sm model 3 lr - 40k features H_20230810141730.html
[NbConvertApp] Converting notebook ./sm model 3 lr - 40k features G.ipynb to html
[NbConvertApp] Writing 1051777 bytes to sm model 3 lr - 40k features G.html
rendered ./sm model 3 lr - 40k features G.ipynb->../jupyter_rendered/./sm model 3 lr - 40k features G_20230810141730.html
[NbConvertApp] Converting notebook ./sm model 3 lr - 40k features F.ipynb to html
[NbConvertApp] Writing 1049718 bytes to sm model 3 lr - 40k features F.html
rendered ./sm model 3 lr - 40k features F.ipynb->../jupyter_rendered/./sm model 3 lr - 40k features F_20230810141730.html
Time taken to render each notebook:
./sm model 3 lr - 40k features baseline.ipynb: 11.311705589294434
./sm model 3 lr - 40k features A.ipynb: 11.42239499092102
./sm model 3 lr - 40k features B.ipynb: 11.018943786621094
./sm model 3 lr - 40k features E.ipynb: 11.14680004119873
./sm model 3 lr - 40k features D.ipynb: 10.84260082244873
./sm model 3 lr - 40k features C.ipynb: 11.489066123962402
./sm model 3 lr - 40k features H.ipynb: 11.388869762420654
./sm model 3 lr - 40k features G.ipynb: 11.564817190170288
./sm model 3 lr - 40k features F.ipynb: 11.413845300674438
(src-yWZ-XGoe) ronie@debian:~/Masters/Thesis - academic internship/code/src$
```

Figure 13 - Output of Notebook Rendering Pipeline.

This will produce HTML files for each notebook in `${PROJ_PATH}/jupyter_rendered/`. The resulting files which are used in the paper are included in the zip artefact.
NB A rendered copy of all models is included in the Appendices.

References

Harazdovskiy, D. (2022) ‘Optimizing massive MongoDB inserts, load 50 million records faster by 33%!’, *Shelf.io Engineering*, 16 December. Available at: https://medium.com/shelf-io-engineering/50-million-records-insert-in-mongodb-using-node-js-5c62b7d7af5a (Accessed: 5 June 2023).

Performance Best Practices: Hardware and OS Configuration | MongoDB Blog (no date) *MongoDB*. Available at: https://www.mongodb.com/blog/post/performance-best-practices-hardware-and-os-configuration (Accessed: 6 July 2023).

Appendix A: Rendered Jupyter Notebooks - Logistic Regression

I. Feature Set Baseline

In [1]:

```
import click
import pandas as pd
import glob
import os
#import sklearn
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report
from sklearn.preprocessing import StandardScaler
from scipy import stats
from sklearn.linear_model import LogisticRegression
import statsmodels.api as sm
import matplotlib.pyplot as plt
from datetime import datetime, date
# qqplot of the data
import seaborn as sns
import math
import numpy as np
import utils

combo_features = ['domain_to_earliest_cert_delta']

path = "../data/"
filename = None

epoch = datetime(1970,1,1)
# open the training data file, from ../data/ for the most recent merged
training data file saved by prepare-training-data-parallel.py
full_path = path + "merged_20230705-104357_training_adorned-engineered.csv"
# get latest file matching the glob
if not filename:
    filename = max(glob.glob(full_path), key=os.path.getctime)
    click.echo(f"Using {filename} as training data")
    df = pd.read_csv(filename, sep=",")

# shuffle the rows
df = df.sample(frac=1, random_state=42).reset_index(drop=True)

click.echo(df.shape)
click.echo(df.columns)

""" # From prepare-training-data-parallel.py:
# Columns:
url
```

```
verification_time
domain
malicious
dateadded
whois_created
soa_timestamp
ctlog_earliest
ctlog_latest
ctlog_wildcard
whois_created_dayofweek,
ctlog_earliest_dayofweek,
domain_to_cert_delta
"""


# Data cleaning - there may be some epoch values in the whois_created
#  & ct_log_earliest columns, so we need to remove those rows

# convert the whois_created and ctlog_earliest, ctlog_latest columns to
datetime

df = df.loc[df["whois_created"] != ""]
df = df.loc[df["ctlog_earliest"] != ""]
df = df.loc[df["ctlog_latest"] != ""]

# some dates are have nanoseconds in them, so we need to remove the
nanosecond part
df["whois_created"] = df["whois_created"].str.split(".", n = 1, expand =
True)[0]
# some dates has additional timezone information, so we need to remove that
df["whois_created"] = df["whois_created"].apply(lambda x: " ".join(
str(x).split(" ")[:2]))


# convert the whois_created and ct_log_earliest columns to datetime
df = df.astype({"whois_created": 'datetime64[ns]', "ctlog_earliest":
'datetime64[ns]', "ctlog_latest": 'datetime64[ns]'})
df = df.loc[df["whois_created"].ne(pd.Timestamp('1970-01-01 00:00:00'))]
df = df.loc[df["ctlog_earliest"].ne(pd.Timestamp('1970-01-01 00:00:00'))]
df = df.loc[df["ctlog_latest"].ne(pd.Timestamp('1970-01-01 00:00:00'))]


# malicious is a bool
df['malicious'] = df['malicious'].astype('bool')
df['domain'] = df['domain'].astype('string')
Using ../data/merged_20230705-104357_training_adorned-engineered.csv as
training data
(35438, 13)
Index(['url', 'verification_time', 'domain', 'malicious', 'dateadded',
```

```
       'whois_created', 'soa_timestamp', 'ctlog_earliest', 'ctlog_latest',
       'ctlog_wildcard', 'whois_created_dayofweek',
'ctlog_earliest_dayofweek',
       'domain_to_cert_delta'],
      dtype='object')
```

In [2]:

```
#########
# Feature engineering
#########

# remove any rows where the whois_created or ctlog_earliest columns are
null
df = df.loc[df["whois_created"].notnull()]
df = df.loc[df["ctlog_earliest"].notnull()]

# if the data is missing the "whois_created_dayofweek" or
"ctlog_earliest_dayofweek" columns, then add them
# whois created day of the week 0 = Monday, 6 = Sunday
df["whois_created_dayofweek"] = df["whois_created"].apply(
        lambda x: x.weekday() if isinstance(x, date) else None
    )

# cert valid day of the week 0 = Monday, 6 = Sunday
df["ctlog_earliest_dayofweek"] = df["ctlog_earliest"].apply(
        lambda x: x.weekday() if isinstance(x, date) else None
    )

df["ctlog_latest_dayofweek"] = df["ctlog_latest"].apply(
        lambda x: x.weekday() if isinstance(x, date) else None
    )

# set data type of the day of week columns to int
df["whois_created_dayofweek"] =
df["whois_created_dayofweek"].astype("int64")
df["ctlog_earliest_dayofweek"] =
df["ctlog_earliest_dayofweek"].astype("int64")
df["ctlog_latest_dayofweek"] = df["ctlog_latest_dayofweek"].astype("int64")

# domain to cert delta
df["domain_to_earliest_cert_delta"] = df.apply(
        lambda x: utils.get_days_delta(
            x["ctlog_earliest"],
            x["whois_created"]
            if x["whois_created"] and x["ctlog_earliest"]
            else None,
        ),
        axis=1,
    )
```

```
# set the data type of the domain_to_cert_delta column to float
df["domain_to_earliest_cert_delta"] =
df["domain_to_earliest_cert_delta"].astype("float64")

# domain to latest cert delta
df["domain_to_latest_cert_delta"] = df.apply(
        lambda x: utils.get_days_delta(
            x["ctlog_latest"],
            x["whois_created"]
            if x["whois_created"] and x["ctlog_latest"]
            else None,
        ),
        axis=1,
    )

# set the data type of the domain_to_cert_delta column to float
df["domain_to_latest_cert_delta"] =
df["domain_to_latest_cert_delta"].astype("float64")

# drop columns we imported that we will never use
df = df.drop(columns=["url", "verification_time", "dateadded",
"soa_timestamp","domain_to_cert_delta"])

click.echo(df.head())
click.echo(df.describe(include='all'))
click.echo(df.dtypes)
                             domain  malicious       whois_created
0  i-db5p-cor001.api.p001.1drv.com      False 2013-08-05 18:33:50  \
4       soundcloud-pax.pandora.com      False 1993-12-28 05:00:00
5         joolcomercializadora.com       True 2023-05-22 14:53:50
6        createpdf-asr.acrobat.com      False 1999-03-16 05:00:00
8                          popt.in      False 2016-05-14 16:58:55

       ctlog_earliest        ctlog_latest  ctlog_wildcard
0 2022-01-24 20:01:58 2023-06-08 20:46:06            True  \
4 2022-05-19 00:00:00 2023-06-19 23:59:59            True
5 2022-04-06 22:23:24 2023-09-22 23:59:59           False
6 2022-09-09 00:00:00 2023-10-10 23:59:59            True
8 2023-01-07 20:36:15 2023-08-15 04:16:52           False

   whois_created_dayofweek  ctlog_earliest_dayofweek
ctlog_latest_dayofweek
0                        0                          0
3  \
4                        1                          3
0
5                        0                          2
4
```

```
6                        1                          4
1
8                        5                          5
1

   domain_to_earliest_cert_delta  domain_to_latest_cert_delta
0                        -3095.0                      -3595.0
4                       -10369.0                     -10766.0
5                          410.0                       -124.0
6                        -8578.0                      -8975.0
8                        -2430.0                      -2649.0
                   domain malicious                  whois_created
count               21549     21549                          21549  \
unique              21536         2                            NaN
top     www.mediafire.com     False                            NaN
freq                    2     11739                            NaN
mean                  NaN       NaN  2012-10-03 12:56:32.335050496
min                   NaN       NaN            1986-01-09 00:00:00
25%                   NaN       NaN            2003-05-25 13:35:05
50%                   NaN       NaN            2015-05-07 23:56:05
75%                   NaN       NaN            2023-03-20 15:03:16
max                   NaN       NaN            2023-07-03 08:21:24
std                   NaN       NaN                            NaN

                       ctlog_earliest                   ctlog_latest
count                           21549                          21549  \
unique                            NaN                            NaN
top                               NaN                            NaN
freq                              NaN                            NaN
mean    2022-09-26 15:45:50.943570432  2023-08-14 17:49:06.400900352
min               2021-11-30 05:24:28            2023-01-01 18:42:11
25%               2022-06-24 13:47:12            2023-07-02 08:11:07
50%               2022-10-18 21:00:14            2023-08-21 21:40:11
75%               2022-12-14 00:00:00            2023-09-21 19:41:38
max               2023-06-28 04:36:22            2023-12-31 23:59:59
std                               NaN                            NaN

       ctlog_wildcard  whois_created_dayofweek  ctlog_earliest_dayofweek
count           21549             21549.000000              21549.000000  \
unique              2                      NaN                       NaN
top             False                      NaN                       NaN
freq            13032                      NaN                       NaN
mean              NaN                 2.332823                  2.399462
min               NaN                 0.000000                  0.000000
25%               NaN                 1.000000                  1.000000
50%               NaN                 2.000000                  2.000000
75%               NaN                 4.000000                  4.000000
max               NaN                 6.000000                  6.000000
std               NaN                 1.775043                  1.897252
```

```
        ctlog_latest_dayofweek  domain_to_earliest_cert_delta
count              21549.000000                   21549.000000  \
unique                      NaN                            NaN
top                         NaN                            NaN
freq                        NaN                            NaN
mean                   2.873080                   -3645.602070
min                    0.000000                  -13445.000000
25%                    1.000000                   -7078.000000
50%                    3.000000                   -2637.000000
75%                    5.000000                      69.000000
max                    6.000000                     524.000000
std                    2.057394                    3790.677119

        domain_to_latest_cert_delta
count                  21549.000000
unique                          NaN
top                             NaN
freq                            NaN
mean                   -3967.678222
min                   -13798.000000
25%                    -7421.000000
50%                    -3009.000000
75%                     -144.000000
max                      135.000000
std                     3852.703681
domain                             string[python]
malicious                                    bool
whois_created                      datetime64[ns]
ctlog_earliest                     datetime64[ns]
ctlog_latest                       datetime64[ns]
ctlog_wildcard                               bool
whois_created_dayofweek                     int64
ctlog_earliest_dayofweek                    int64
ctlog_latest_dayofweek                      int64
domain_to_earliest_cert_delta             float64
domain_to_latest_cert_delta               float64
dtype: object
```

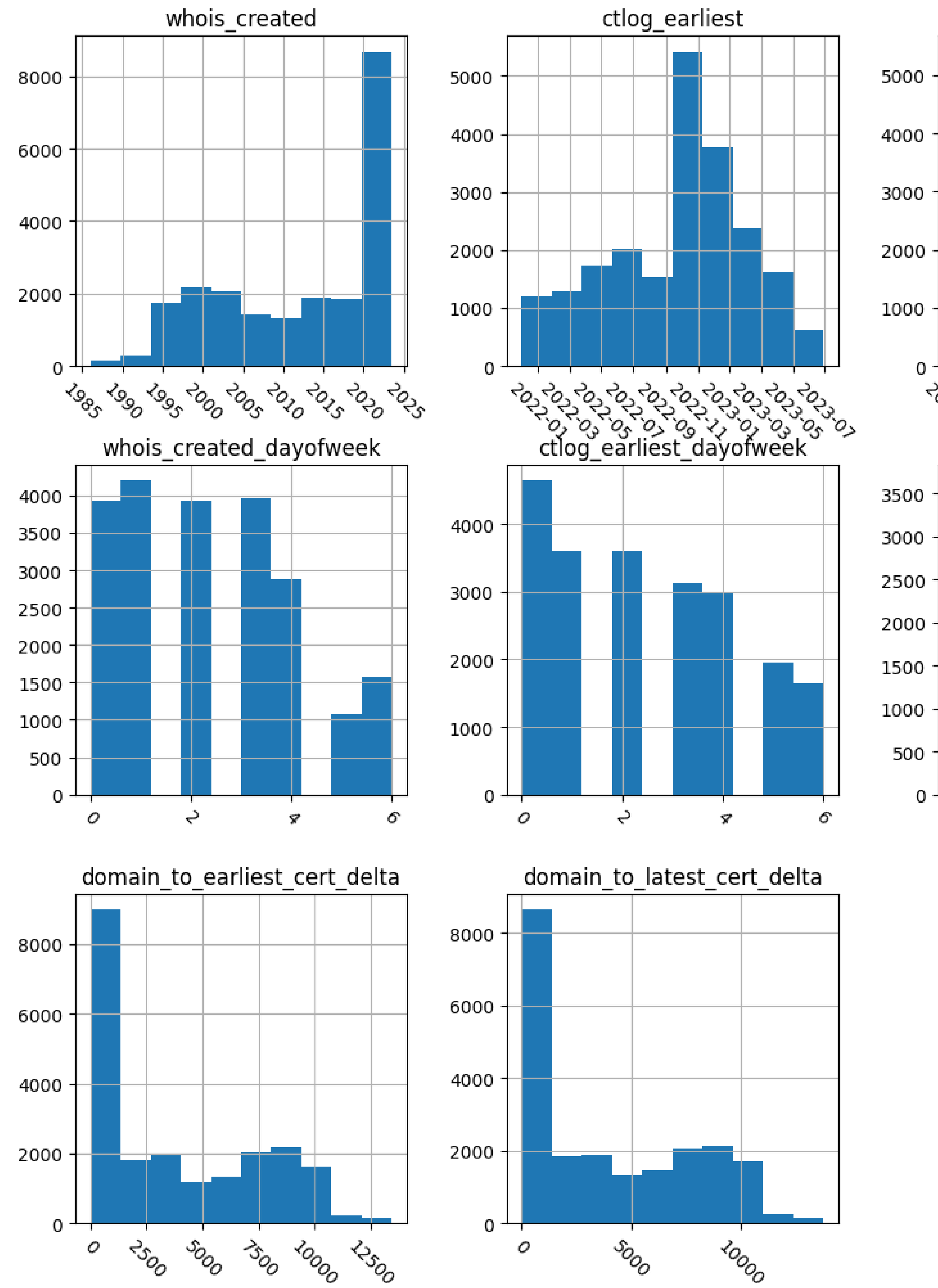
In [3]:

```
# absolute value of the domain_to_cert_delta
df["domain_to_earliest_cert_delta"] =
abs(df["domain_to_earliest_cert_delta"])
df["domain_to_latest_cert_delta"] = abs(df["domain_to_latest_cert_delta"])

df.hist(figsize=(12,12), xrot=-45)

col_index = df.columns.get_loc("whois_created_dayofweek")
df["whois_created_dow_sin"] = df.apply(lambda row:
math.sin(360/7*(row[col_index])),axis=1)
```

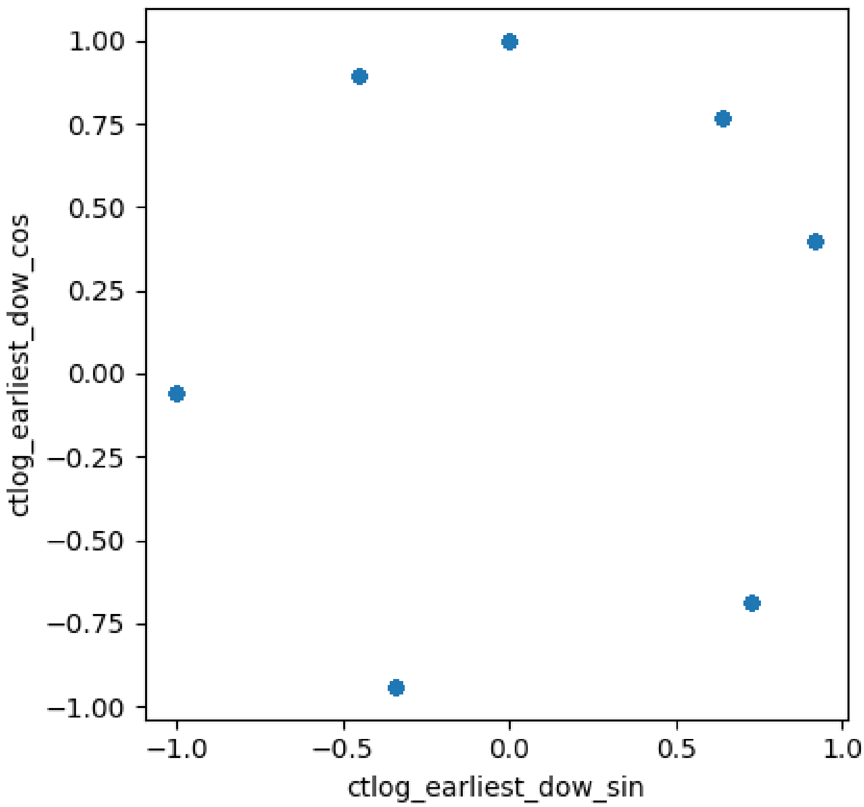
```
df["whois_created_dow_cos"] = df.apply(lambda row:
math.cos(360/7*(row[col_index])),axis=1)

col_index = df.columns.get_loc("ctlog_earliest_dayofweek")
df["ctlog_earliest_dow_sin"] = df.apply(lambda row:
math.sin(360/7*(row[col_index])),axis=1)
df["ctlog_earliest_dow_cos"] = df.apply(lambda row:
math.cos(360/7*(row[col_index])),axis=1)

col_index = df.columns.get_loc("ctlog_latest_dayofweek")
df["ctlog_latest_dow_sin"] = df.apply(lambda row:
math.sin(360/7*(row[col_index])),axis=1)
df["ctlog_latest_dow_cos"] = df.apply(lambda row:
math.cos(360/7*(row[col_index])),axis=1)


df.plot.scatter(x="ctlog_earliest_dow_sin",
y="ctlog_earliest_dow_cos").set_aspect('equal')
df.plot.scatter(x="whois_created_dow_sin",
y="whois_created_dow_cos").set_aspect('equal')
df.plot.scatter(x="ctlog_latest_dow_sin",
y="ctlog_latest_dow_cos").set_aspect('equal')

"""
# add one hot encode ctlog_earliest_not_before_dayofweek
df = pd.get_dummies(
        df,
        columns=["ctlog_earliest_dayofweek"],
        prefix=["ctlog_earliest_dow"],
    )
# add one hot encode whois_created_dayofweek to columns
df = pd.get_dummies(
        df, columns=["whois_created_dayofweek"], prefix="whois_dow"
    )
    """

# Summary statistics
click.echo(df.describe(include='all'))
                   domain malicious                 whois_created
count               21549     21549                         21549  \
unique              21536         2                           NaN
top      www.mediafire.com     False                           NaN
freq                    2     11739                           NaN
mean                  NaN       NaN  2012-10-03 12:56:32.335050496
min                   NaN       NaN            1986-01-09 00:00:00
25%                   NaN       NaN            2003-05-25 13:35:05
50%                   NaN       NaN            2015-05-07 23:56:05
75%                   NaN       NaN            2023-03-20 15:03:16
max                   NaN       NaN            2023-07-03 08:21:24
```

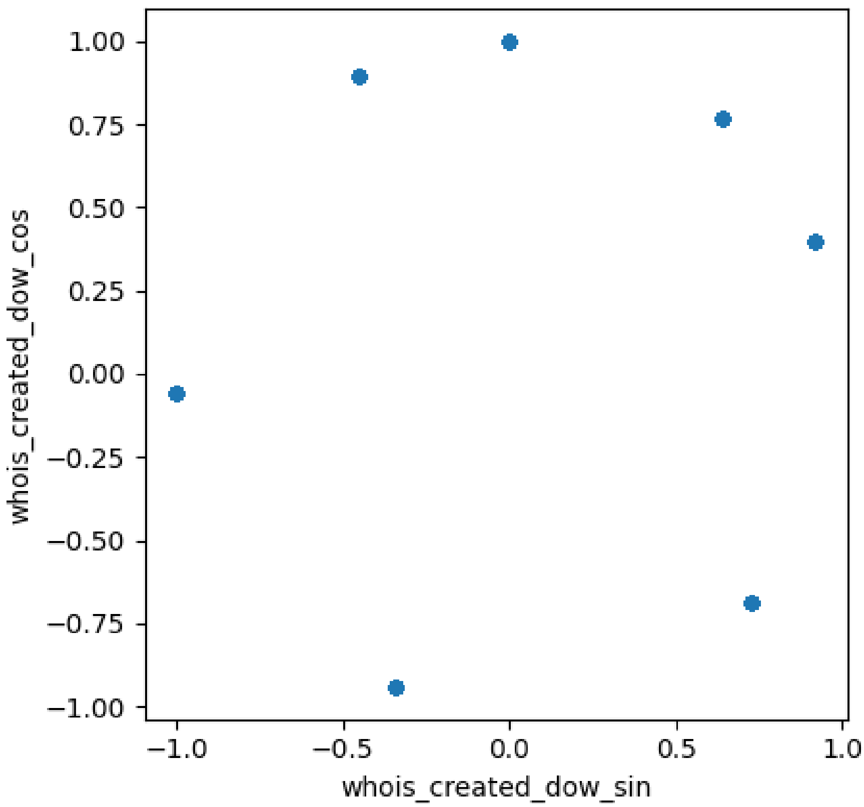
```
std                  NaN       NaN                              NaN

                   ctlog_earliest                 ctlog_latest
count                       21549                        21549  \
unique                        NaN                          NaN
top                           NaN                          NaN
freq                          NaN                          NaN
mean    2022-09-26 15:45:50.943570432  2023-08-14 17:49:06.400900352
min           2021-11-30 05:24:28          2023-01-01 18:42:11
25%           2022-06-24 13:47:12          2023-07-02 08:11:07
50%           2022-10-18 21:00:14          2023-08-21 21:40:11
75%           2022-12-14 00:00:00          2023-09-21 19:41:38
max           2023-06-28 04:36:22          2023-12-31 23:59:59
std                           NaN                          NaN

       ctlog_wildcard  whois_created_dayofweek  ctlog_earliest_dayofweek
count           21549             21549.000000              21549.000000  \
unique              2                      NaN                       NaN
top             False                      NaN                       NaN
freq            13032                      NaN                       NaN
mean              NaN                 2.332823                  2.399462
min               NaN                 0.000000                  0.000000
25%               NaN                 1.000000                  1.000000
50%               NaN                 2.000000                  2.000000
75%               NaN                 4.000000                  4.000000
max               NaN                 6.000000                  6.000000
std               NaN                 1.775043                  1.897252

        ctlog_latest_dayofweek  domain_to_earliest_cert_delta
count             21549.000000                   21549.000000  \
unique                     NaN                            NaN
top                        NaN                            NaN
freq                       NaN                            NaN
mean                  2.873080                    3742.948397
min                   0.000000                       0.000000
25%                   1.000000                     181.000000
50%                   3.000000                    2637.000000
75%                   5.000000                    7078.000000
max                   6.000000                   13445.000000
std                   2.057394                    3694.584062

        domain_to_latest_cert_delta  whois_created_dow_sin
count                  21549.000000           21549.000000  \
unique                          NaN                    NaN
top                             NaN                    NaN
freq                            NaN                    NaN
mean                    3969.491206               0.140419
min                        0.000000              -0.998199
25%                      144.000000              -0.340712
```

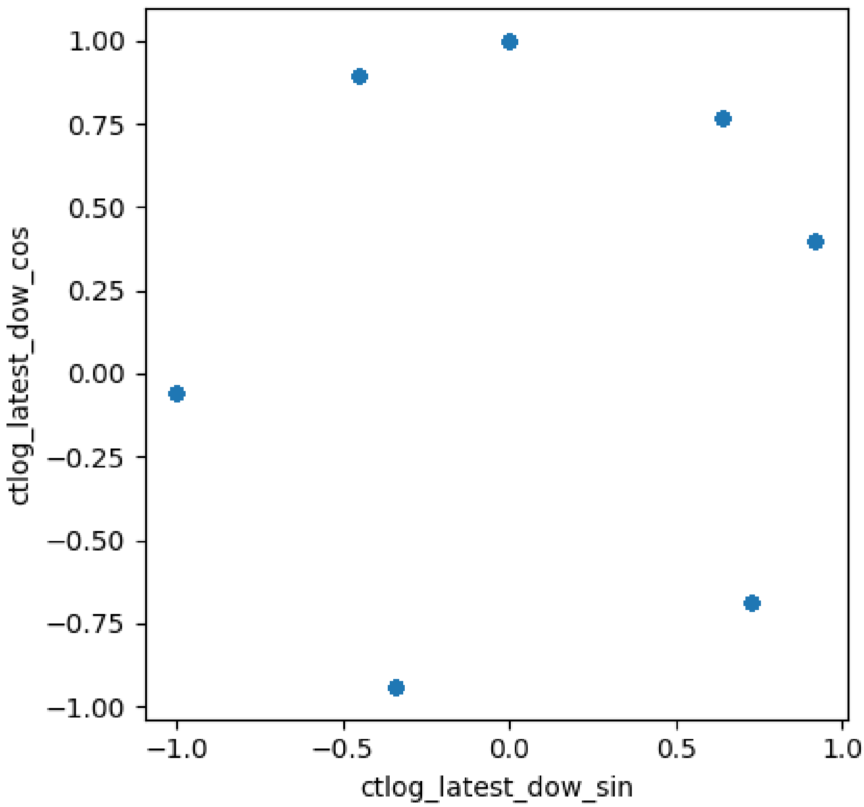
```
50%                     3009.000000               0.000000
75%                     7421.000000               0.728010
max                    13798.000000               0.918032
std                     3850.835626               0.659922

        whois_created_dow_cos  ctlog_earliest_dow_sin
ctlog_earliest_dow_cos
count            21549.000000            21549.000000
21549.000000  \
unique                    NaN                     NaN
NaN
top                       NaN                     NaN
NaN
freq                      NaN                     NaN
NaN
mean                 0.054288                0.095357
0.161451
min                 -0.940168               -0.998199               -
0.940168
25%                 -0.685567               -0.340712               -
0.685567
50%                  0.396506                0.000000
0.396506
75%                  0.767830                0.728010
0.892589
max                  1.000000                0.918032
1.000000
std                  0.736128                0.651782
0.734891

        ctlog_latest_dow_sin  ctlog_latest_dow_cos
count           21549.000000          21549.000000
unique                   NaN                   NaN
top                      NaN                   NaN
freq                     NaN                   NaN
mean                0.096253              0.255578
min                -0.998199             -0.940168
25%                -0.450871             -0.685567
50%                 0.000000              0.396506
75%                 0.728010              0.892589
max                 0.918032              1.000000
std                 0.651597              0.707728
```



whois_created
0
2000
4000
6000
8000
1985
1990
1995
2000
2005
2010
2015
2020
2025
ctlog_earliest
0
1000
2000
3000
4000
5000
2022-01
2022-03
2022-05
2022-07
2022-09
2022-11
2023-01
2023-03
2023-05
2023-07
whois_created_dayofweek
0
500
1000
1500
2000
2500
3000
3500
4000
0
2
4
6
ctlog_earliest_dayofweek
0
1000
2000
3000
4000
0
2
4
6
domain_to_earliest_cert_delta
0
2000
4000
6000
8000
0
2500
5000
7500
10000
12500
domain_to_latest_cert_delta
0
2000
4000
6000
8000
0
5000
10000


1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−1.0
−0.5
0.0
0.5
1.0
ctlog_earliest_dow_sin
ctlog_earliest_dow_cos

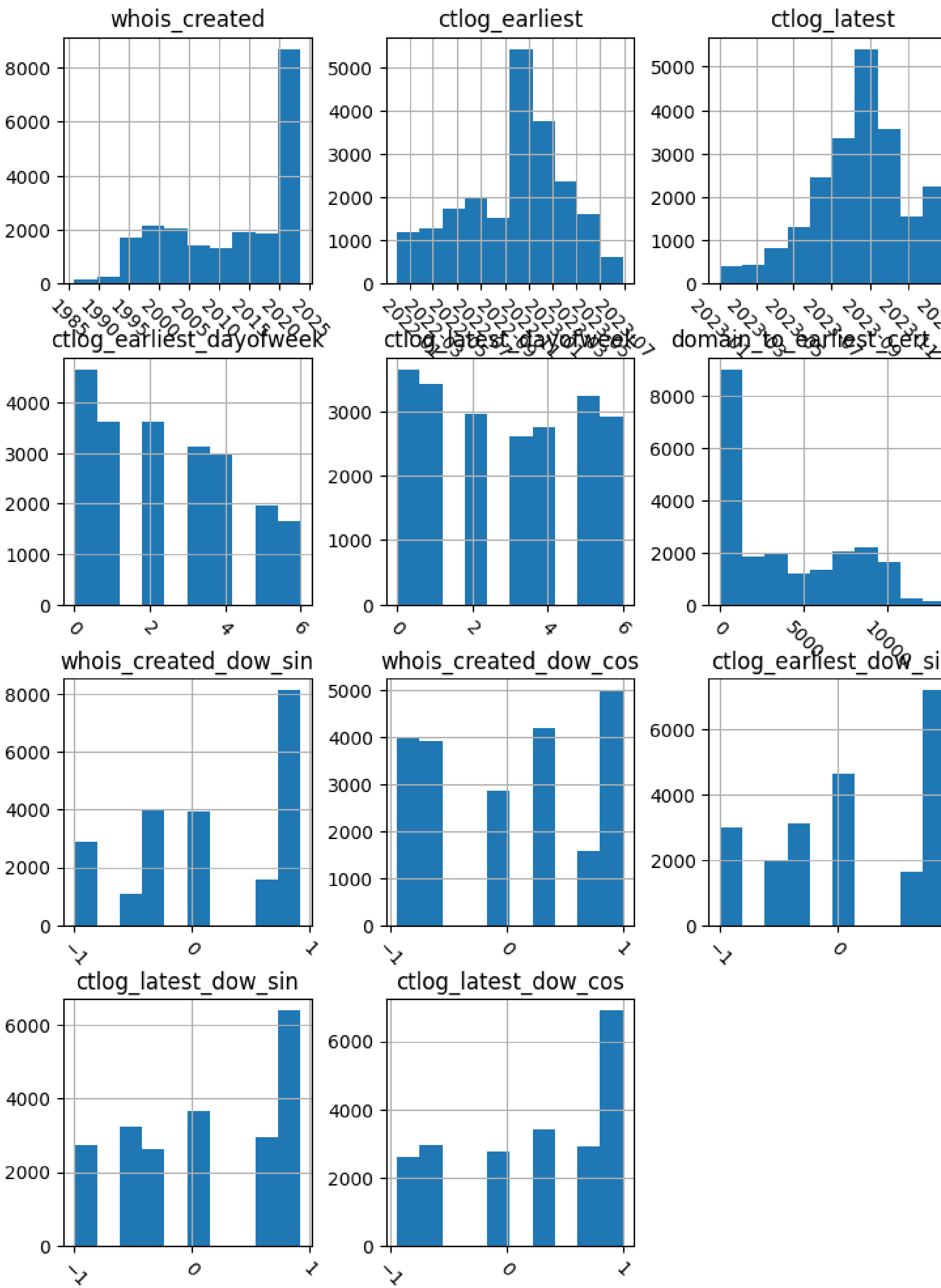

1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−1.0
−0.5
0.0
0.5
1.0
whois_created_dow_sin
whois_created_dow_cos



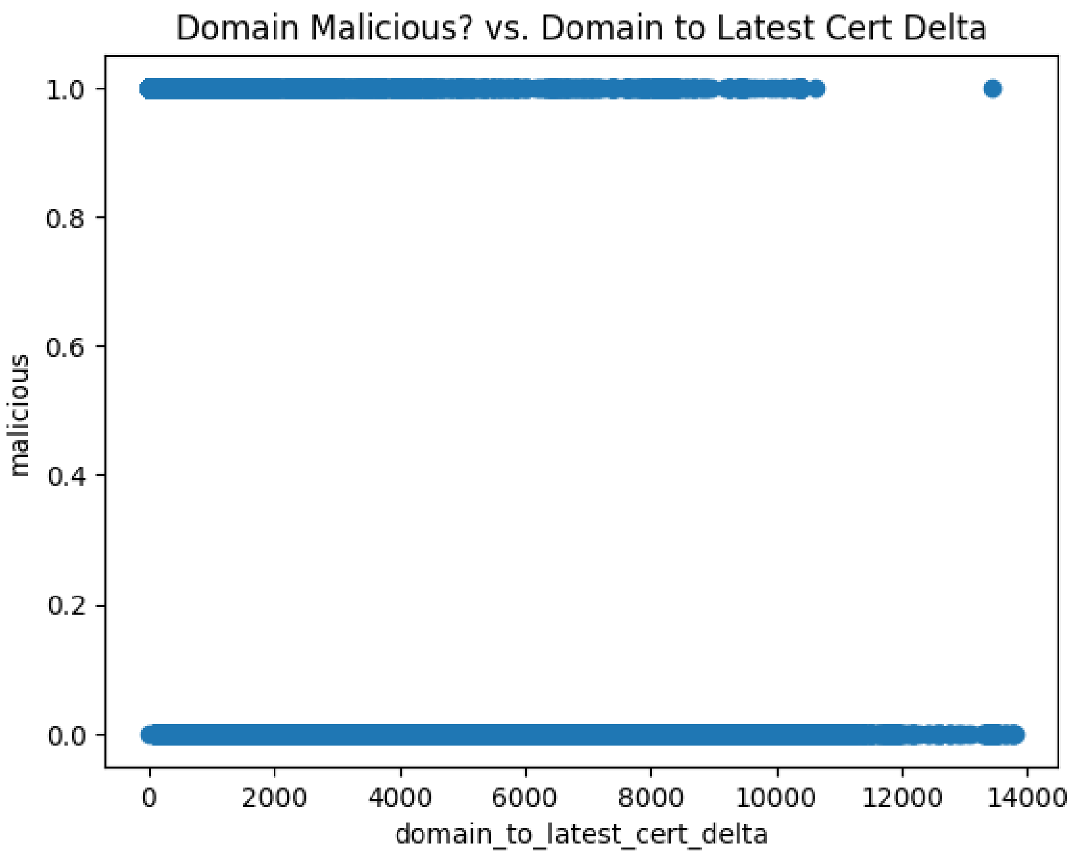
In [4]:

```
# Plot histograms
df.hist(figsize=(12,12), xrot=-45)

# Create a scatter plot of the data, showing malicious and benign domains
# plot the data as a scatter plot
plt.scatter(df["domain_to_earliest_cert_delta"], df["malicious"])
plt.xlabel("domain_to_earliest_cert_delta")
plt.ylabel("malicious")
plt.title("Domain Malicious? vs. Domain to Earliest Cert Delta")
plt.show()
# and for the latest
plt.scatter(df["domain_to_latest_cert_delta"], df["malicious"])
plt.xlabel("domain_to_latest_cert_delta")
plt.ylabel("malicious")
plt.title("Domain Malicious? vs. Domain to Latest Cert Delta")
plt.show()



click.echo(df.head())

# print a count of malicious and benign values
click.echo(df["malicious"].value_counts())
# deduplicate any rows, there shouldn't be any, but just in case
df = df.drop_duplicates()
click.echo(df["malicious"].value_counts())
```

```
click.echo(df.head())

X = df.drop(["malicious","domain", "ctlog_earliest", "ctlog_latest",
"whois_created", "whois_created_dayofweek", "ctlog_earliest_dayofweek"],
axis=1)
X = df.filter(combo_features)
y = df.filter(["malicious"])


click.echo(X.describe())
```



whois_created
ctlog_earliest
ctlog_latest
ctlog_earliest_dayofweek
ctlog_latest_dayofweek
domain_to_earliest_cert_
whois_created_dow_sin
whois_created_dow_cos
ctlog_earliest_dow_si
ctlog_latest_dow_sin
ctlog_latest_dow_cos




```
                              domain  malicious       whois_created
0  i-db5p-cor001.api.p001.1drv.com      False 2013-08-05 18:33:50   \
4       soundcloud-pax.pandora.com      False 1993-12-28 05:00:00
5         joolcomercializadora.com       True 2023-05-22 14:53:50
6        createpdf-asr.acrobat.com      False 1999-03-16 05:00:00
8                          popt.in      False 2016-05-14 16:58:55

       ctlog_earliest        ctlog_latest  ctlog_wildcard
0 2022-01-24 20:01:58 2023-06-08 20:46:06            True   \
4 2022-05-19 00:00:00 2023-06-19 23:59:59            True
5 2022-04-06 22:23:24 2023-09-22 23:59:59           False
6 2022-09-09 00:00:00 2023-10-10 23:59:59            True
8 2023-01-07 20:36:15 2023-08-15 04:16:52           False

   whois_created_dayofweek  ctlog_earliest_dayofweek
ctlog_latest_dayofweek
0                        0                          0
3   \
4                        1                          3
0
5                        0                          2
4
6                        1                          4
1
8                        5                          5
1
```

```
   domain_to_earliest_cert_delta  domain_to_latest_cert_delta
0                         3095.0                       3595.0  \
4                        10369.0                      10766.0
5                          410.0                        124.0
6                         8578.0                       8975.0
8                         2430.0                       2649.0

   whois_created_dow_sin  whois_created_dow_cos  ctlog_earliest_dow_sin
0               0.000000               1.000000                0.000000  \
4               0.918032               0.396506               -0.340712
5               0.000000               1.000000                0.728010
6               0.918032               0.396506               -0.998199
8              -0.450871               0.892589               -0.450871

   ctlog_earliest_dow_cos  ctlog_latest_dow_sin  ctlog_latest_dow_cos
0                1.000000             -0.340712             -0.940168
4               -0.940168              0.000000              1.000000
5               -0.685567             -0.998199             -0.059997
6               -0.059997              0.918032              0.396506
8                0.892589              0.918032              0.396506
malicious
False    11739
True      9810
Name: count, dtype: int64
malicious
False    11739
True      9810
Name: count, dtype: int64
                            domain  malicious       whois_created
0  i-db5p-cor001.api.p001.1drv.com      False 2013-08-05 18:33:50  \
4       soundcloud-pax.pandora.com      False 1993-12-28 05:00:00
5         joolcomercializadora.com       True 2023-05-22 14:53:50
6        createpdf-asr.acrobat.com      False 1999-03-16 05:00:00
8                          popt.in      False 2016-05-14 16:58:55

       ctlog_earliest        ctlog_latest  ctlog_wildcard
0 2022-01-24 20:01:58 2023-06-08 20:46:06            True  \
4 2022-05-19 00:00:00 2023-06-19 23:59:59            True
5 2022-04-06 22:23:24 2023-09-22 23:59:59           False
6 2022-09-09 00:00:00 2023-10-10 23:59:59            True
8 2023-01-07 20:36:15 2023-08-15 04:16:52           False

   whois_created_dayofweek  ctlog_earliest_dayofweek
ctlog_latest_dayofweek
0                        0                         0
3  \
4                        1                         3
0
```

```
5                         0                          2
4
6                         1                          4
1
8                         5                          5
1

   domain_to_earliest_cert_delta  domain_to_latest_cert_delta
0                         3095.0                       3595.0  \
4                        10369.0                      10766.0
5                          410.0                        124.0
6                         8578.0                       8975.0
8                         2430.0                       2649.0

   whois_created_dow_sin  whois_created_dow_cos  ctlog_earliest_dow_sin
0               0.000000               1.000000                0.000000  \
4               0.918032               0.396506               -0.340712
5               0.000000               1.000000                0.728010
6               0.918032               0.396506               -0.998199
8              -0.450871               0.892589               -0.450871

   ctlog_earliest_dow_cos  ctlog_latest_dow_sin  ctlog_latest_dow_cos
0                1.000000             -0.340712             -0.940168
4               -0.940168              0.000000              1.000000
5               -0.685567             -0.998199             -0.059997
6               -0.059997              0.918032              0.396506
8                0.892589              0.918032              0.396506
       domain_to_earliest_cert_delta
count                   21549.000000
mean                     3742.948397
std                      3694.584062
min                         0.000000
25%                       181.000000
50%                      2637.000000
75%                      7078.000000
max                     13445.000000
```

In [5]:

```
# convert y (malicious) to 1/0 int
y = y.astype('int')
# convert X["ctlog_wildcard"]  to 1/0 int
if "ctlog_wildcard" in X.columns:
    X["ctlog_wildcard"] = X["ctlog_wildcard"].astype('int')

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

X_train = sm.add_constant(X_train)
X_test = sm.add_constant(X_test)
```

```
smfit = sm.Logit(y_train,X_train).fit()

smfit.summary()
Optimization terminated successfully.
         Current function value: 0.372985
         Iterations 7
```

Out[5]:

Logit Regression Results

Dep. Variable:	malicious	**No. Observations:**	17239
Model:	Logit	**Df Residuals:**	17237
Method:	MLE	**Df Model:**	1
Date:	Tue, 08 Aug 2023	**Pseudo R-squ.:**	0.4590
Time:	19:08:30	**Log-Likelihood:**	-6429.9
converged:	True	**LL-Null:**	-11885.
Covariance Type:	nonrobust	**LLR p-value:**	0.000

	coef	**std err**	**z**	**P>|z|**	**[0.025**	**0.975]**
const	1.8025	0.030	59.114	0.000	1.743	1.862
domain_to_earliest_cert_delta	-0.0007	1.01e-05	-67.857	0.000	-0.001	-0.001

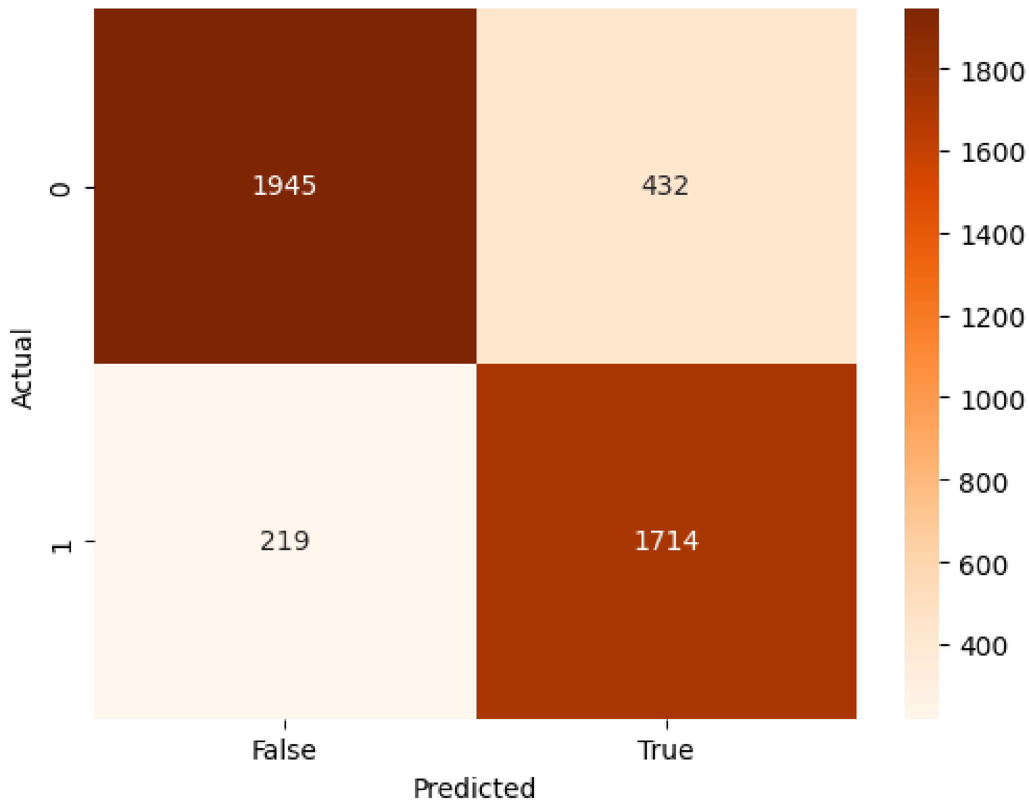
In [6]:

```
# Predict the malicious column using the test data
#add the incepts


y_predicted = smfit.predict(X_test)

# Present the results in a confusion matrix
confusion_matrix = confusion_matrix(y_test, y_predicted.round())
click.echo(confusion_matrix)

click.echo("Classification report:")
click.echo(classification_report(y_test, y_predicted.round()))

# Heatmap of confusion matrix
y_predicted

threshold = 0.5
y_predicted = [y > threshold for y in y_predicted]
data = {'Actual':    y_test.values.flatten(),
        'Predicted': y_predicted
        }

# Generate a confusion matrix and heatmap to evaluate the Type I and Type
II errors/ FP/FN etc.
df = pd.DataFrame(data, columns=['Actual','Predicted'])
confusion_matrix = pd.crosstab(df['Actual'], df['Predicted'],
rownames=['Actual'], colnames=['Predicted'])
fig = sns.heatmap(confusion_matrix, annot=True, cmap='Oranges', fmt='g')
fig
```

```
[[1945  432]
 [ 219 1714]]
Classification report:
              precision    recall  f1-score   support

           0       0.90      0.82      0.86      2377
           1       0.80      0.89      0.84      1933

    accuracy                           0.85      4310
   macro avg       0.85      0.85      0.85      4310
weighted avg       0.85      0.85      0.85      4310
```

Out[6]:

```
<Axes: xlabel='Predicted', ylabel='Actual'>
```



II. Feature Set A

In [1]:

```
import click
import pandas as pd
import glob
import os
#import sklearn
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report
from sklearn.preprocessing import StandardScaler
from scipy import stats
from sklearn.linear_model import LogisticRegression
import statsmodels.api as sm
import matplotlib.pyplot as plt
from datetime import datetime, date
# qqplot of the data
import seaborn as sns
import math
import numpy as np
import utils

combo_features = ['domain_to_earliest_cert_delta',
'ctlog_earliest_dow_sin', 'ctlog_earliest_dow_cos']

path = "../data/"
filename = None

epoch = datetime(1970,1,1)
# open the training data file, from ../data/ for the most recent merged
training data file saved by prepare-training-data-parallel.py
full_path = path + "merged_20230705-104357_training_adorned-engineered.csv"
# get latest file matching the glob
if not filename:
    filename = max(glob.glob(full_path), key=os.path.getctime)
    click.echo(f"Using {filename} as training data")
    df = pd.read_csv(filename, sep=",")

# shuffle the rows
df = df.sample(frac=1, random_state=42).reset_index(drop=True)

click.echo(df.shape)
click.echo(df.columns)

""" # From prepare-training-data-parallel.py:
# Columns:
url
verification_time
domain
```

```
malicious
dateadded
whois_created
soa_timestamp
ctlog_earliest
ctlog_latest
ctlog_wildcard
whois_created_dayofweek,
ctlog_earliest_dayofweek,
domain_to_cert_delta
"""


# Data cleaning - there may be some epoch values in the whois_created
#  & ct_log_earliest columns, so we need to remove those rows

# convert the whois_created and ctlog_earliest, ctlog_latest columns to
datetime

df = df.loc[df["whois_created"] != ""]
df = df.loc[df["ctlog_earliest"] != ""]
df = df.loc[df["ctlog_latest"] != ""]

# some dates are have nanoseconds in them, so we need to remove the
nanosecond part
df["whois_created"] = df["whois_created"].str.split(".", n = 1, expand =
True)[0]
# some dates has additional timezone information, so we need to remove that
df["whois_created"] = df["whois_created"].apply(lambda x: " ".join(
str(x).split(" ")[:2]))


# convert the whois_created and ct_log_earliest columns to datetime
df = df.astype({"whois_created": 'datetime64[ns]', "ctlog_earliest":
'datetime64[ns]', "ctlog_latest": 'datetime64[ns]'})
df = df.loc[df["whois_created"].ne(pd.Timestamp('1970-01-01 00:00:00'))]
df = df.loc[df["ctlog_earliest"].ne(pd.Timestamp('1970-01-01 00:00:00'))]
df = df.loc[df["ctlog_latest"].ne(pd.Timestamp('1970-01-01 00:00:00'))]


# malicious is a bool
df['malicious'] = df['malicious'].astype('bool')
df['domain'] = df['domain'].astype('string')
Using ../data/merged_20230705-104357_training_adorned-engineered.csv as
training data
(35438, 13)
Index(['url', 'verification_time', 'domain', 'malicious', 'dateadded',
       'whois_created', 'soa_timestamp', 'ctlog_earliest', 'ctlog_latest',
```

```
       'ctlog_wildcard', 'whois_created_dayofweek',
'ctlog_earliest_dayofweek',
       'domain_to_cert_delta'],
      dtype='object')
```

In [2]:

```
#########
# Feature engineering
#########

# remove any rows where the whois_created or ctlog_earliest columns are
null
df = df.loc[df["whois_created"].notnull()]
df = df.loc[df["ctlog_earliest"].notnull()]

# if the data is missing the "whois_created_dayofweek" or
"ctlog_earliest_dayofweek" columns, then add them
# whois created day of the week 0 = Monday, 6 = Sunday
df["whois_created_dayofweek"] = df["whois_created"].apply(
        lambda x: x.weekday() if isinstance(x, date) else None
    )

# cert valid day of the week 0 = Monday, 6 = Sunday
df["ctlog_earliest_dayofweek"] = df["ctlog_earliest"].apply(
        lambda x: x.weekday() if isinstance(x, date) else None
    )

df["ctlog_latest_dayofweek"] = df["ctlog_latest"].apply(
        lambda x: x.weekday() if isinstance(x, date) else None
    )

# set data type of the day of week columns to int
df["whois_created_dayofweek"] =
df["whois_created_dayofweek"].astype("int64")
df["ctlog_earliest_dayofweek"] =
df["ctlog_earliest_dayofweek"].astype("int64")
df["ctlog_latest_dayofweek"] = df["ctlog_latest_dayofweek"].astype("int64")

# domain to cert delta
df["domain_to_earliest_cert_delta"] = df.apply(
        lambda x: utils.get_days_delta(
            x["ctlog_earliest"],
            x["whois_created"]
            if x["whois_created"] and x["ctlog_earliest"]
            else None,
        ),
        axis=1,
    )

# set the data type of the domain_to_cert_delta column to float
```

```
df["domain_to_earliest_cert_delta"] =
df["domain_to_earliest_cert_delta"].astype("float64")

# domain to latest cert delta
df["domain_to_latest_cert_delta"] = df.apply(
        lambda x: utils.get_days_delta(
            x["ctlog_latest"],
            x["whois_created"]
            if x["whois_created"] and x["ctlog_latest"]
            else None,
        ),
        axis=1,
    )

# set the data type of the domain_to_cert_delta column to float
df["domain_to_latest_cert_delta"] =
df["domain_to_latest_cert_delta"].astype("float64")

# drop columns we imported that we will never use
df = df.drop(columns=["url", "verification_time", "dateadded",
"soa_timestamp","domain_to_cert_delta"])

click.echo(df.head())
click.echo(df.describe(include='all'))
click.echo(df.dtypes)
                            domain  malicious       whois_created
0  i-db5p-cor001.api.p001.1drv.com      False 2013-08-05 18:33:50  \
4       soundcloud-pax.pandora.com      False 1993-12-28 05:00:00
5         joolcomercializadora.com       True 2023-05-22 14:53:50
6        createpdf-asr.acrobat.com      False 1999-03-16 05:00:00
8                          popt.in      False 2016-05-14 16:58:55

       ctlog_earliest        ctlog_latest  ctlog_wildcard
0 2022-01-24 20:01:58 2023-06-08 20:46:06            True  \
4 2022-05-19 00:00:00 2023-06-19 23:59:59            True
5 2022-04-06 22:23:24 2023-09-22 23:59:59           False
6 2022-09-09 00:00:00 2023-10-10 23:59:59            True
8 2023-01-07 20:36:15 2023-08-15 04:16:52           False

   whois_created_dayofweek  ctlog_earliest_dayofweek
ctlog_latest_dayofweek
0                        0                          0
3  \
4                        1                          3
0
5                        0                          2
4
6                        1                          4
1
```

```
8                        5                          5
1

   domain_to_earliest_cert_delta  domain_to_latest_cert_delta
0                        -3095.0                      -3595.0
4                       -10369.0                     -10766.0
5                          410.0                       -124.0
6                        -8578.0                      -8975.0
8                        -2430.0                      -2649.0
                    domain malicious                 whois_created
count               21549     21549                          21549  \
unique              21536         2                            NaN
top     www.mediafire.com     False                            NaN
freq                    2     11739                            NaN
mean                  NaN       NaN  2012-10-03 12:56:32.335050496
min                   NaN       NaN            1986-01-09 00:00:00
25%                   NaN       NaN            2003-05-25 13:35:05
50%                   NaN       NaN            2015-05-07 23:56:05
75%                   NaN       NaN            2023-03-20 15:03:16
max                   NaN       NaN            2023-07-03 08:21:24
std                   NaN       NaN                            NaN

                        ctlog_earliest                   ctlog_latest
count                            21549                          21549  \
unique                             NaN                            NaN
top                                NaN                            NaN
freq                               NaN                            NaN
mean     2022-09-26 15:45:50.943570432  2023-08-14 17:49:06.400900352
min                2021-11-30 05:24:28            2023-01-01 18:42:11
25%                2022-06-24 13:47:12            2023-07-02 08:11:07
50%                2022-10-18 21:00:14            2023-08-21 21:40:11
75%                2022-12-14 00:00:00            2023-09-21 19:41:38
max                2023-06-28 04:36:22            2023-12-31 23:59:59
std                                NaN                            NaN

       ctlog_wildcard  whois_created_dayofweek  ctlog_earliest_dayofweek
count           21549             21549.000000              21549.000000  \
unique              2                      NaN                       NaN
top             False                      NaN                       NaN
freq            13032                      NaN                       NaN
mean              NaN                 2.332823                  2.399462
min               NaN                 0.000000                  0.000000
25%               NaN                 1.000000                  1.000000
50%               NaN                 2.000000                  2.000000
75%               NaN                 4.000000                  4.000000
max               NaN                 6.000000                  6.000000
std               NaN                 1.775043                  1.897252

        ctlog_latest_dayofweek  domain_to_earliest_cert_delta
```

```
count             21549.000000                21549.000000  \
unique                     NaN                         NaN
top                        NaN                         NaN
freq                       NaN                         NaN
mean                  2.873080                -3645.602070
min                   0.000000               -13445.000000
25%                   1.000000                -7078.000000
50%                   3.000000                -2637.000000
75%                   5.000000                   69.000000
max                   6.000000                  524.000000
std                   2.057394                 3790.677119

        domain_to_latest_cert_delta
count                  21549.000000
unique                          NaN
top                             NaN
freq                            NaN
mean                   -3967.678222
min                   -13798.000000
25%                    -7421.000000
50%                    -3009.000000
75%                     -144.000000
max                      135.000000
std                     3852.703681
domain                             string[python]
malicious                                    bool
whois_created                      datetime64[ns]
ctlog_earliest                     datetime64[ns]
ctlog_latest                       datetime64[ns]
ctlog_wildcard                               bool
whois_created_dayofweek                     int64
ctlog_earliest_dayofweek                    int64
ctlog_latest_dayofweek                      int64
domain_to_earliest_cert_delta             float64
domain_to_latest_cert_delta               float64
dtype: object
```

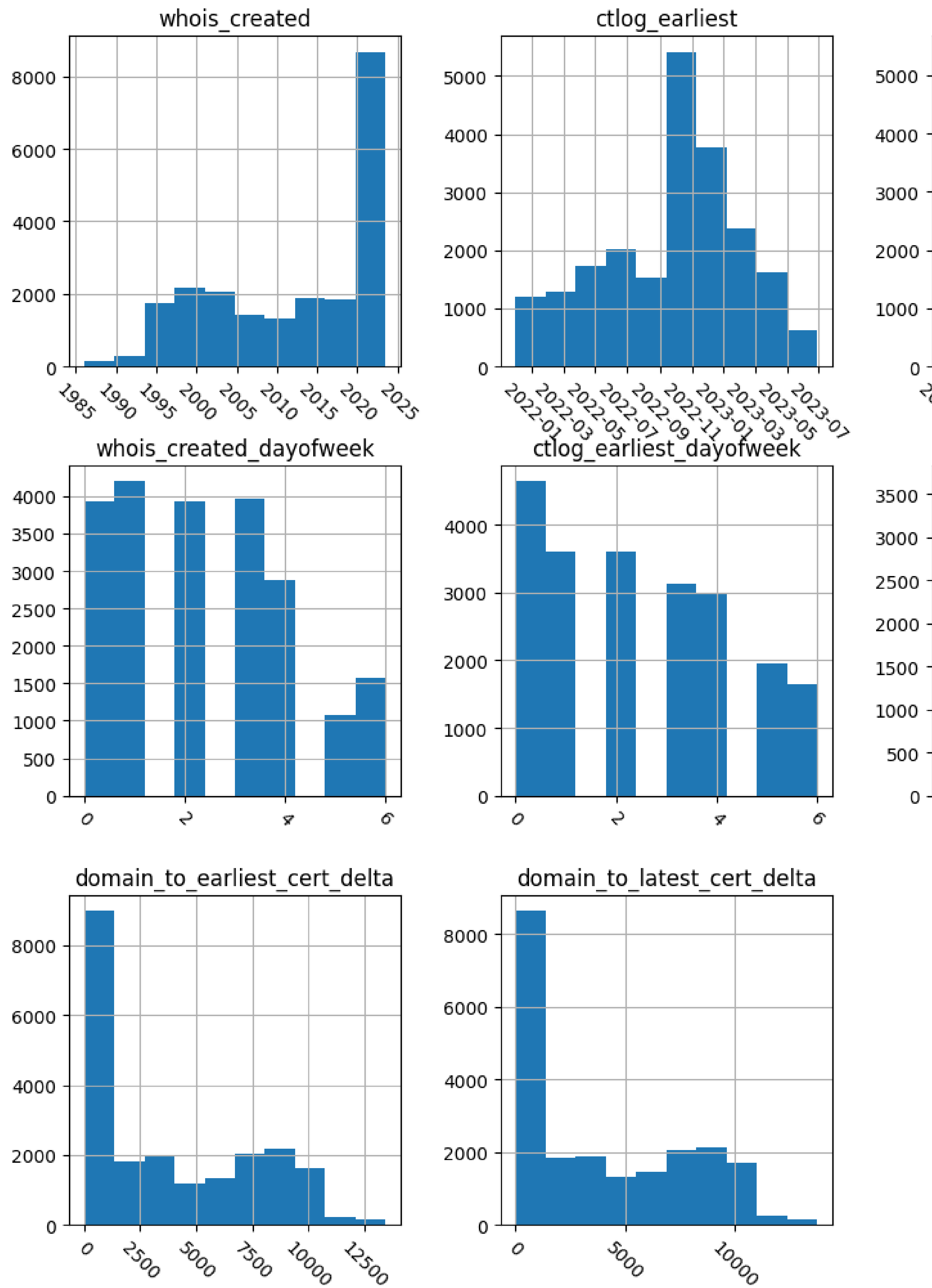
In [3]:

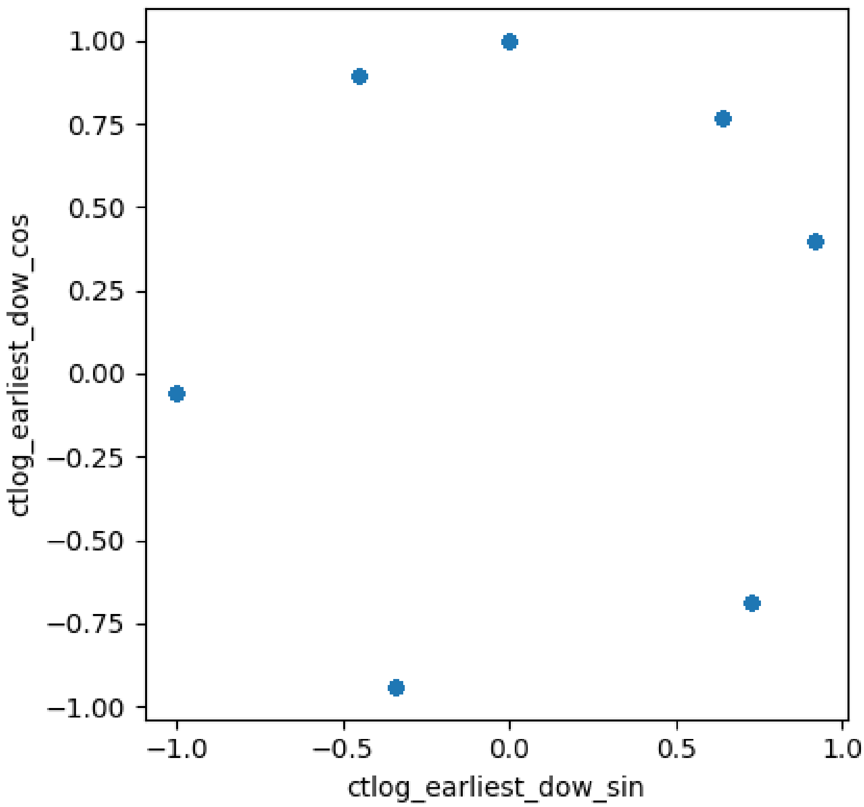
```
# absolute value of the domain_to_cert_delta
df["domain_to_earliest_cert_delta"] =
abs(df["domain_to_earliest_cert_delta"])
df["domain_to_latest_cert_delta"] = abs(df["domain_to_latest_cert_delta"])

df.hist(figsize=(12,12), xrot=-45)

col_index = df.columns.get_loc("whois_created_dayofweek")
df["whois_created_dow_sin"] = df.apply(lambda row:
math.sin(360/7*(row[col_index])),axis=1)
df["whois_created_dow_cos"] = df.apply(lambda row:
math.cos(360/7*(row[col_index])),axis=1)
```

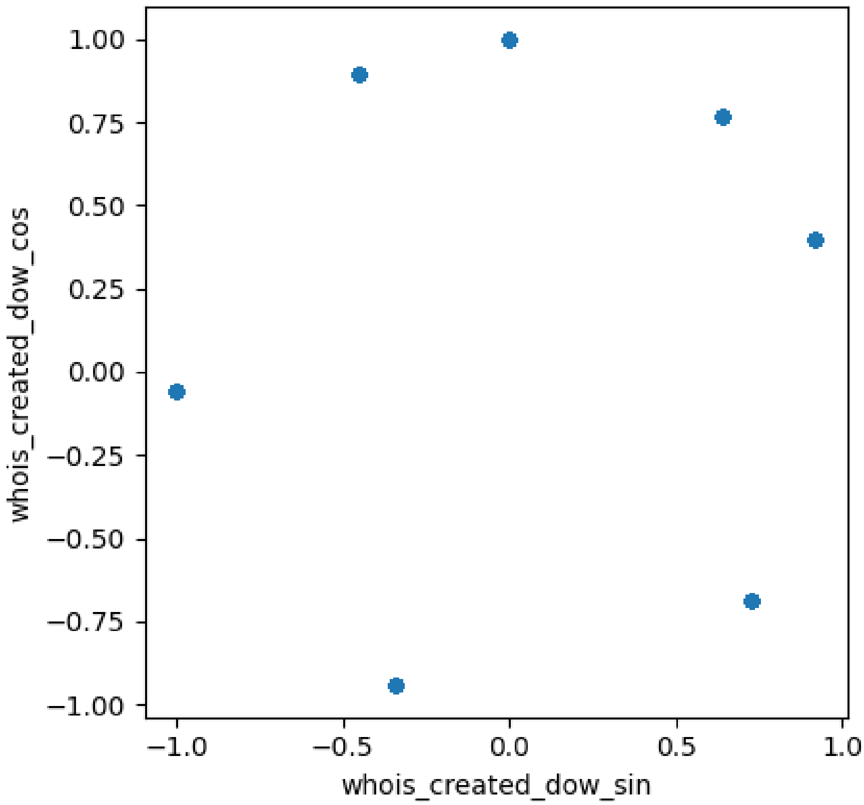
```
col_index = df.columns.get_loc("ctlog_earliest_dayofweek")
df["ctlog_earliest_dow_sin"] = df.apply(lambda row:
math.sin(360/7*(row[col_index])),axis=1)
df["ctlog_earliest_dow_cos"] = df.apply(lambda row:
math.cos(360/7*(row[col_index])),axis=1)

col_index = df.columns.get_loc("ctlog_latest_dayofweek")
df["ctlog_latest_dow_sin"] = df.apply(lambda row:
math.sin(360/7*(row[col_index])),axis=1)
df["ctlog_latest_dow_cos"] = df.apply(lambda row:
math.cos(360/7*(row[col_index])),axis=1)


df.plot.scatter(x="ctlog_earliest_dow_sin",
y="ctlog_earliest_dow_cos").set_aspect('equal')
df.plot.scatter(x="whois_created_dow_sin",
y="whois_created_dow_cos").set_aspect('equal')
df.plot.scatter(x="ctlog_latest_dow_sin",
y="ctlog_latest_dow_cos").set_aspect('equal')

"""
# add one hot encode ctlog_earliest_not_before_dayofweek
df = pd.get_dummies(
        df,
        columns=["ctlog_earliest_dayofweek"],
        prefix=["ctlog_earliest_dow"],
    )
# add one hot encode whois_created_dayofweek to columns
df = pd.get_dummies(
        df, columns=["whois_created_dayofweek"], prefix="whois_dow"
    )
    """

# Summary statistics
click.echo(df.describe(include='all'))
                   domain malicious                    whois_created
count               21549     21549                            21549  \
unique              21536         2                              NaN
top     www.mediafire.com     False                              NaN
freq                    2     11739                              NaN
mean                  NaN       NaN  2012-10-03 12:56:32.335050496
min                   NaN       NaN            1986-01-09 00:00:00
25%                   NaN       NaN            2003-05-25 13:35:05
50%                   NaN       NaN            2015-05-07 23:56:05
75%                   NaN       NaN            2023-03-20 15:03:16
max                   NaN       NaN            2023-07-03 08:21:24
std                   NaN       NaN                              NaN
```

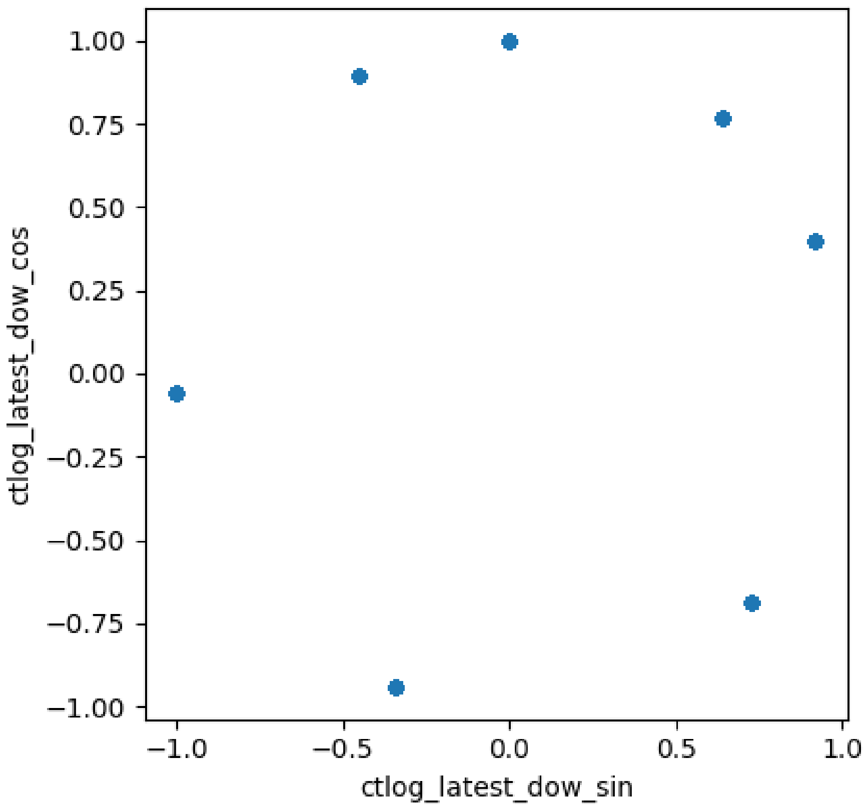
```
                  ctlog_earliest                   ctlog_latest
count                      21549                          21549  \
unique                       NaN                            NaN
top                          NaN                            NaN
freq                         NaN                            NaN
mean    2022-09-26 15:45:50.943570432  2023-08-14 17:49:06.400900352
min              2021-11-30 05:24:28            2023-01-01 18:42:11
25%              2022-06-24 13:47:12            2023-07-02 08:11:07
50%              2022-10-18 21:00:14            2023-08-21 21:40:11
75%              2022-12-14 00:00:00            2023-09-21 19:41:38
max              2023-06-28 04:36:22            2023-12-31 23:59:59
std                          NaN                            NaN

       ctlog_wildcard  whois_created_dayofweek  ctlog_earliest_dayofweek
count           21549             21549.000000              21549.000000  \
unique              2                      NaN                       NaN
top             False                      NaN                       NaN
freq            13032                      NaN                       NaN
mean              NaN                 2.332823                  2.399462
min               NaN                 0.000000                  0.000000
25%               NaN                 1.000000                  1.000000
50%               NaN                 2.000000                  2.000000
75%               NaN                 4.000000                  4.000000
max               NaN                 6.000000                  6.000000
std               NaN                 1.775043                  1.897252

        ctlog_latest_dayofweek  domain_to_earliest_cert_delta
count             21549.000000                   21549.000000  \
unique                     NaN                            NaN
top                        NaN                            NaN
freq                       NaN                            NaN
mean                  2.873080                    3742.948397
min                   0.000000                       0.000000
25%                   1.000000                     181.000000
50%                   3.000000                    2637.000000
75%                   5.000000                    7078.000000
max                   6.000000                   13445.000000
std                   2.057394                    3694.584062

        domain_to_latest_cert_delta  whois_created_dow_sin
count                  21549.000000           21549.000000  \
unique                          NaN                    NaN
top                             NaN                    NaN
freq                            NaN                    NaN
mean                    3969.491206               0.140419
min                        0.000000              -0.998199
25%                      144.000000              -0.340712
50%                     3009.000000               0.000000
75%                     7421.000000               0.728010
```

```
max                    13798.000000               0.918032
std                     3850.835626               0.659922

        whois_created_dow_cos  ctlog_earliest_dow_sin
ctlog_earliest_dow_cos
count             21549.000000            21549.000000
21549.000000  \
unique                     NaN                     NaN
NaN
top                        NaN                     NaN
NaN
freq                       NaN                     NaN
NaN
mean                  0.054288                0.095357
0.161451
min                  -0.940168               -0.998199                -
0.940168
25%                  -0.685567               -0.340712                -
0.685567
50%                   0.396506                0.000000
0.396506
75%                   0.767830                0.728010
0.892589
max                   1.000000                0.918032
1.000000
std                   0.736128                0.651782
0.734891

        ctlog_latest_dow_sin  ctlog_latest_dow_cos
count            21549.000000          21549.000000
unique                    NaN                   NaN
top                       NaN                   NaN
freq                      NaN                   NaN
mean                 0.096253              0.255578
min                 -0.998199             -0.940168
25%                 -0.450871             -0.685567
50%                  0.000000              0.396506
75%                  0.728010              0.892589
max                  0.918032              1.000000
std                  0.651597              0.707728
```


whois_created
0
2000
4000
6000
8000
1985
1990
1995
2000
2005
2010
2015
2020
2025
ctlog_earliest
0
1000
2000
3000
4000
5000
2022-01
2022-03
2022-05
2022-07
2022-09
2022-11
2023-01
2023-03
2023-05
2023-07
whois_created_dayofweek
0
500
1000
1500
2000
2500
3000
3500
4000
0
2
4
6
ctlog_earliest_dayofweek
0
1000
2000
3000
4000
0
2
4
6
domain_to_earliest_cert_delta
0
2000
4000
6000
8000
0
2500
5000
7500
10000
12500
domain_to_latest_cert_delta
0
2000
4000
6000
8000
0
5000
10000


1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
ctlog_earliest_dow_cos
−1.0
−0.5
0.0
0.5
1.0
ctlog_earliest_dow_sin

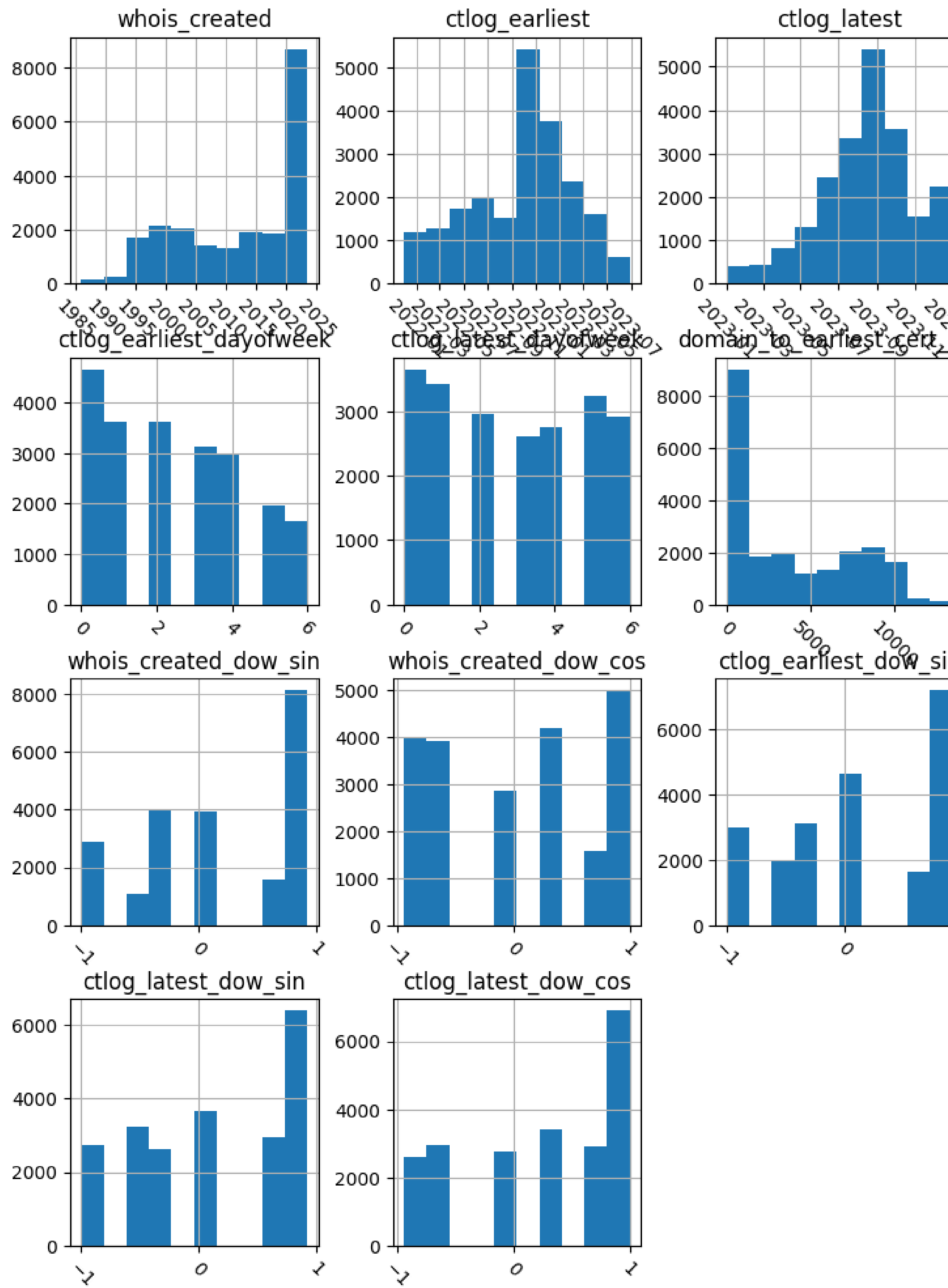

1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
whois_created_dow_cos
−1.0
−0.5
0.0
0.5
1.0
whois_created_dow_sin



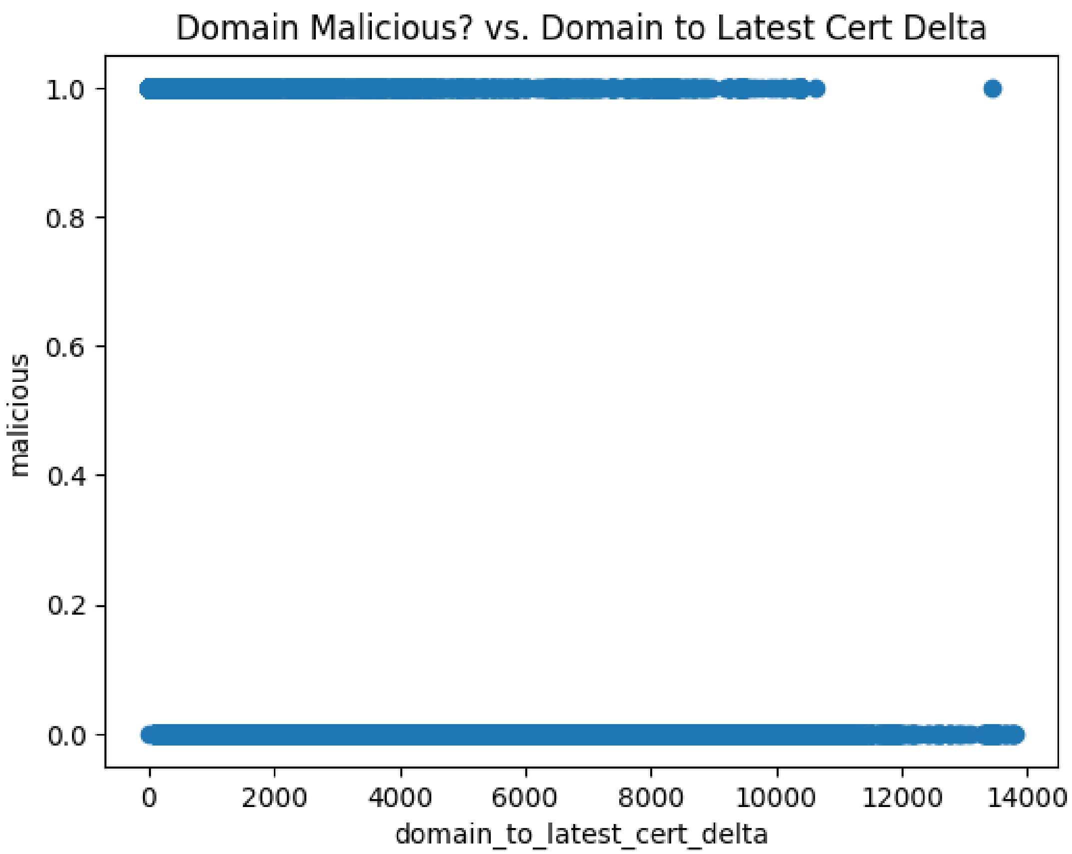
In [4]:

```
# Plot histograms
df.hist(figsize=(12,12), xrot=-45)

# Create a scatter plot of the data, showing malicious and benign domains
# plot the data as a scatter plot
plt.scatter(df["domain_to_earliest_cert_delta"], df["malicious"])
plt.xlabel("domain_to_earliest_cert_delta")
plt.ylabel("malicious")
plt.title("Domain Malicious? vs. Domain to Earliest Cert Delta")
plt.show()
# and for the latest
plt.scatter(df["domain_to_latest_cert_delta"], df["malicious"])
plt.xlabel("domain_to_latest_cert_delta")
plt.ylabel("malicious")
plt.title("Domain Malicious? vs. Domain to Latest Cert Delta")
plt.show()



click.echo(df.head())

# print a count of malicious and benign values
click.echo(df["malicious"].value_counts())
# deduplicate any rows, there shouldn't be any, but just in case
df = df.drop_duplicates()
click.echo(df["malicious"].value_counts())
```

```
click.echo(df.head())

X = df.drop(["malicious","domain", "ctlog_earliest", "ctlog_latest",
"whois_created", "whois_created_dayofweek", "ctlog_earliest_dayofweek"],
axis=1)
X = df.filter(combo_features)
y = df.filter(["malicious"])


click.echo(X.describe())
```



whois_created
ctlog_earliest
ctlog_latest
ctlog_earliest_dayofweek
ctlog_latest_dayofweek
whois_created_dow_sin
whois_created_dow_cos
ctlog_latest_dow_sin
ctlog_latest_dow_cos




```
                            domain  malicious        whois_created
0  i-db5p-cor001.api.p001.1drv.com      False 2013-08-05 18:33:50   \
4       soundcloud-pax.pandora.com      False 1993-12-28 05:00:00
5         joolcomercializadora.com       True 2023-05-22 14:53:50
6        createpdf-asr.acrobat.com      False 1999-03-16 05:00:00
8                          popt.in      False 2016-05-14 16:58:55

        ctlog_earliest        ctlog_latest  ctlog_wildcard  \
0 2022-01-24 20:01:58 2023-06-08 20:46:06            True   \
4 2022-05-19 00:00:00 2023-06-19 23:59:59            True
5 2022-04-06 22:23:24 2023-09-22 23:59:59           False
6 2022-09-09 00:00:00 2023-10-10 23:59:59            True
8 2023-01-07 20:36:15 2023-08-15 04:16:52           False

   whois_created_dayofweek  ctlog_earliest_dayofweek
ctlog_latest_dayofweek
0                        0                          0
3  \
4                        1                          3
0
5                        0                          2
4
6                        1                          4
1
8                        5                          5
1
```

```
   domain_to_earliest_cert_delta  domain_to_latest_cert_delta
0                         3095.0                       3595.0  \
4                        10369.0                      10766.0
5                          410.0                        124.0
6                         8578.0                       8975.0
8                         2430.0                       2649.0

   whois_created_dow_sin  whois_created_dow_cos  ctlog_earliest_dow_sin
0               0.000000               1.000000                0.000000  \
4               0.918032               0.396506               -0.340712
5               0.000000               1.000000                0.728010
6               0.918032               0.396506               -0.998199
8              -0.450871               0.892589               -0.450871

   ctlog_earliest_dow_cos  ctlog_latest_dow_sin  ctlog_latest_dow_cos
0                1.000000             -0.340712             -0.940168
4               -0.940168              0.000000              1.000000
5               -0.685567             -0.998199             -0.059997
6               -0.059997              0.918032              0.396506
8                0.892589              0.918032              0.396506
malicious
False    11739
True      9810
Name: count, dtype: int64
malicious
False    11739
True      9810
Name: count, dtype: int64
                            domain  malicious       whois_created
0  i-db5p-cor001.api.p001.1drv.com      False 2013-08-05 18:33:50  \
4       soundcloud-pax.pandora.com      False 1993-12-28 05:00:00
5         joolcomercializadora.com       True 2023-05-22 14:53:50
6        createpdf-asr.acrobat.com      False 1999-03-16 05:00:00
8                          popt.in      False 2016-05-14 16:58:55

       ctlog_earliest        ctlog_latest  ctlog_wildcard
0 2022-01-24 20:01:58 2023-06-08 20:46:06            True  \
4 2022-05-19 00:00:00 2023-06-19 23:59:59            True
5 2022-04-06 22:23:24 2023-09-22 23:59:59           False
6 2022-09-09 00:00:00 2023-10-10 23:59:59            True
8 2023-01-07 20:36:15 2023-08-15 04:16:52           False

   whois_created_dayofweek  ctlog_earliest_dayofweek
ctlog_latest_dayofweek
0                        0                         0
3  \
4                        1                         3
0
```

```
5                    0                    2
4
6                    1                    4
1
8                    5                    5
1

   domain_to_earliest_cert_delta  domain_to_latest_cert_delta
0                         3095.0                       3595.0  \
4                        10369.0                      10766.0
5                          410.0                        124.0
6                         8578.0                       8975.0
8                         2430.0                       2649.0

   whois_created_dow_sin  whois_created_dow_cos  ctlog_earliest_dow_sin
0               0.000000               1.000000                0.000000  \
4               0.918032               0.396506               -0.340712
5               0.000000               1.000000                0.728010
6               0.918032               0.396506               -0.998199
8              -0.450871               0.892589               -0.450871

   ctlog_earliest_dow_cos  ctlog_latest_dow_sin  ctlog_latest_dow_cos
0                1.000000             -0.340712             -0.940168
4               -0.940168              0.000000              1.000000
5               -0.685567             -0.998199             -0.059997
6               -0.059997              0.918032              0.396506
8                0.892589              0.918032              0.396506
       domain_to_earliest_cert_delta  ctlog_earliest_dow_sin
count                   21549.000000            21549.000000  \
mean                     3742.948397                0.095357
std                      3694.584062                0.651782
min                         0.000000               -0.998199
25%                       181.000000               -0.340712
50%                      2637.000000                0.000000
75%                      7078.000000                0.728010
max                     13445.000000                0.918032

       ctlog_earliest_dow_cos
count            21549.000000
mean                 0.161451
std                  0.734891
min                 -0.940168
25%                 -0.685567
50%                  0.396506
75%                  0.892589
max                  1.000000
```

In [5]:

```
# convert y (malicious) to 1/0 int
y = y.astype('int')
```

```
# convert X["ctlog_wildcard"]  to 1/0 int
if "ctlog_wildcard" in X.columns:
    X["ctlog_wildcard"] = X["ctlog_wildcard"].astype('int')

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

X_train = sm.add_constant(X_train)
X_test = sm.add_constant(X_test)

smfit = sm.Logit(y_train,X_train).fit()

smfit.summary()
Optimization terminated successfully.
         Current function value: 0.372073
         Iterations 7
```

Out[5]:

Logit Regression Results

Dep. Variable:	malicious	**No. Observations:**	17239
Model:	Logit	**Df Residuals:**	17235
Method:	MLE	**Df Model:**	3
Date:	Tue, 08 Aug 2023	**Pseudo R-squ.:**	0.4603
Time:	19:08:41	**Log-Likelihood:**	-6414.2
converged:	True	**LL-Null:**	-11885.
Covariance Type:	nonrobust	**LLR p-value:**	0.000

	coef	**std err**	**z**	**P>|z|**	**[0.025**	**0.975]**
const	1.8153	0.031	58.571	0.000	1.755	1.876
domain_to_earliest_cert_delta	-0.0007	1.02e-05	-67.559	0.000	-0.001	-0.001
ctlog_earliest_dow_sin	0.1281	0.034	3.760	0.000	0.061	0.195
ctlog_earliest_dow_cos	-0.1308	0.030	-4.317	0.000	-0.190	-0.071

In [6]:

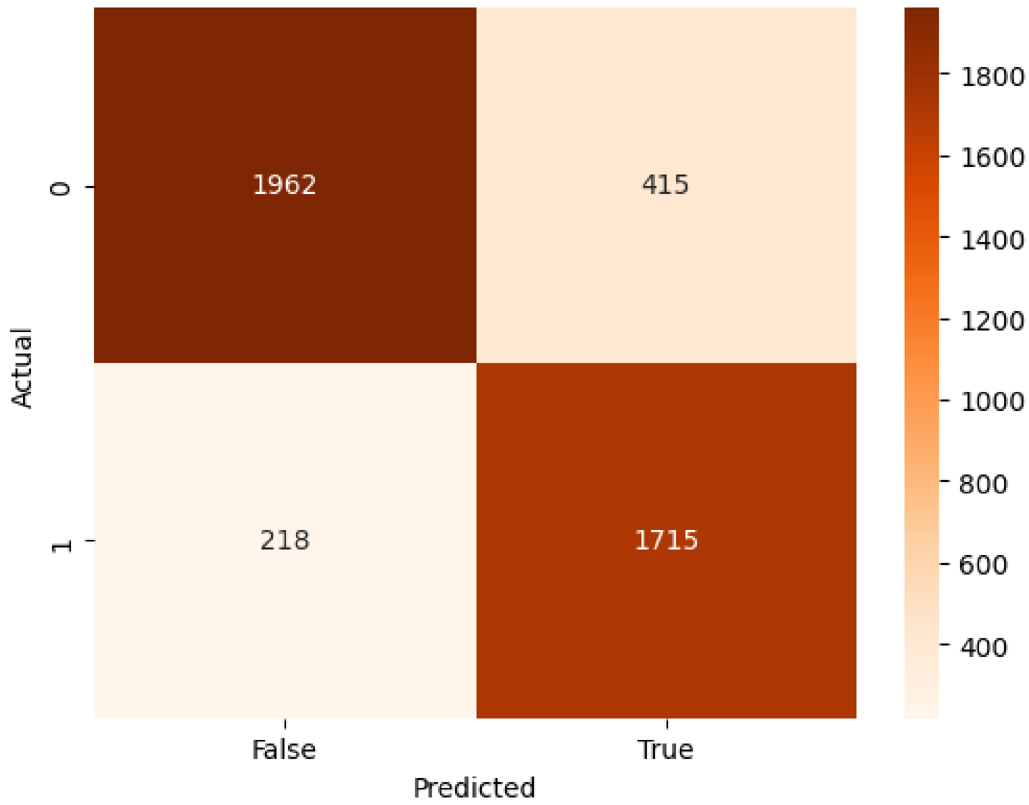
```
# Predict the malicious column using the test data
#add the incepts


y_predicted = smfit.predict(X_test)

# Present the results in a confusion matrix
confusion_matrix = confusion_matrix(y_test, y_predicted.round())
click.echo(confusion_matrix)

click.echo("Classification report:")
click.echo(classification_report(y_test, y_predicted.round()))


# Heatmap of confusion matrix
y_predicted
```

```
threshold = 0.5
y_predicted = [y > threshold for y in y_predicted]
data = {'Actual':    y_test.values.flatten(),
        'Predicted': y_predicted
        }

# Generate a confusion matrix and heatmap to evaluate the Type I and Type
II errors/ FP/FN etc.
df = pd.DataFrame(data, columns=['Actual','Predicted'])
confusion_matrix = pd.crosstab(df['Actual'], df['Predicted'],
rownames=['Actual'], colnames=['Predicted'])
fig = sns.heatmap(confusion_matrix, annot=True, cmap='Oranges', fmt='g')
fig
```

```
[[1962  415]
 [ 218 1715]]
Classification report:
              precision    recall  f1-score   support

           0       0.90      0.83      0.86      2377
           1       0.81      0.89      0.84      1933

    accuracy                           0.85      4310
   macro avg       0.85      0.86      0.85      4310
weighted avg       0.86      0.85      0.85      4310
```

Out[6]:

```
<Axes: xlabel='Predicted', ylabel='Actual'>
```



III. Feature Set B

In [1]:

```
import click
import pandas as pd
import glob
import os
#import sklearn
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report
from sklearn.preprocessing import StandardScaler
from scipy import stats
from sklearn.linear_model import LogisticRegression
import statsmodels.api as sm
import matplotlib.pyplot as plt
from datetime import datetime, date
# qqplot of the data
import seaborn as sns
import math
import numpy as np
import utils

combo_features =     ['domain_to_earliest_cert_delta', 'ctlog_wildcard']

path = "../data/"
filename = None

epoch = datetime(1970,1,1)
# open the training data file, from ../data/ for the most recent merged
training data file saved by prepare-training-data-parallel.py
full_path = path + "merged_20230705-104357_training_adorned-engineered.csv"
# get latest file matching the glob
if not filename:
    filename = max(glob.glob(full_path), key=os.path.getctime)
    click.echo(f"Using {filename} as training data")
    df = pd.read_csv(filename, sep=",")

# shuffle the rows
df = df.sample(frac=1, random_state=42).reset_index(drop=True)

click.echo(df.shape)
click.echo(df.columns)

""" # From prepare-training-data-parallel.py:
# Columns:
url
verification_time
domain
malicious
```

```
dateadded
whois_created
soa_timestamp
ctlog_earliest
ctlog_latest
ctlog_wildcard
whois_created_dayofweek,
ctlog_earliest_dayofweek,
domain_to_cert_delta
"""


# Data cleaning - there may be some epoch values in the whois_created
#  & ct_log_earliest columns, so we need to remove those rows

# convert the whois_created and ctlog_earliest, ctlog_latest columns to
datetime

df = df.loc[df["whois_created"] != ""]
df = df.loc[df["ctlog_earliest"] != ""]
df = df.loc[df["ctlog_latest"] != ""]

# some dates are have nanoseconds in them, so we need to remove the
nanosecond part
df["whois_created"] = df["whois_created"].str.split(".", n = 1, expand =
True)[0]
# some dates has additional timezone information, so we need to remove that
df["whois_created"] = df["whois_created"].apply(lambda x: " ".join(
str(x).split(" ")[:2]))


# convert the whois_created and ct_log_earliest columns to datetime
df = df.astype({"whois_created": 'datetime64[ns]', "ctlog_earliest":
'datetime64[ns]', "ctlog_latest": 'datetime64[ns]'})
df = df.loc[df["whois_created"].ne(pd.Timestamp('1970-01-01 00:00:00'))]
df = df.loc[df["ctlog_earliest"].ne(pd.Timestamp('1970-01-01 00:00:00'))]
df = df.loc[df["ctlog_latest"].ne(pd.Timestamp('1970-01-01 00:00:00'))]


# malicious is a bool
df['malicious'] = df['malicious'].astype('bool')
df['domain'] = df['domain'].astype('string')
Using ../data/merged_20230705-104357_training_adorned-engineered.csv as
training data
(35438, 13)
Index(['url', 'verification_time', 'domain', 'malicious', 'dateadded',
       'whois_created', 'soa_timestamp', 'ctlog_earliest', 'ctlog_latest',
       'ctlog_wildcard', 'whois_created_dayofweek',
'ctlog_earliest_dayofweek',
```

```
       'domain_to_cert_delta'],
      dtype='object')
```

In [2]:

```
#########
# Feature engineering
#########

# remove any rows where the whois_created or ctlog_earliest columns are
null
df = df.loc[df["whois_created"].notnull()]
df = df.loc[df["ctlog_earliest"].notnull()]

# if the data is missing the "whois_created_dayofweek" or
"ctlog_earliest_dayofweek" columns, then add them
# whois created day of the week 0 = Monday, 6 = Sunday
df["whois_created_dayofweek"] = df["whois_created"].apply(
        lambda x: x.weekday() if isinstance(x, date) else None
    )

# cert valid day of the week 0 = Monday, 6 = Sunday
df["ctlog_earliest_dayofweek"] = df["ctlog_earliest"].apply(
        lambda x: x.weekday() if isinstance(x, date) else None
    )

df["ctlog_latest_dayofweek"] = df["ctlog_latest"].apply(
        lambda x: x.weekday() if isinstance(x, date) else None
    )

# set data type of the day of week columns to int
df["whois_created_dayofweek"] =
df["whois_created_dayofweek"].astype("int64")
df["ctlog_earliest_dayofweek"] =
df["ctlog_earliest_dayofweek"].astype("int64")
df["ctlog_latest_dayofweek"] = df["ctlog_latest_dayofweek"].astype("int64")

# domain to cert delta
df["domain_to_earliest_cert_delta"] = df.apply(
        lambda x: utils.get_days_delta(
            x["ctlog_earliest"],
            x["whois_created"]
            if x["whois_created"] and x["ctlog_earliest"]
            else None,
        ),
        axis=1,
    )

# set the data type of the domain_to_cert_delta column to float
df["domain_to_earliest_cert_delta"] =
df["domain_to_earliest_cert_delta"].astype("float64")
```

```
# domain to latest cert delta
df["domain_to_latest_cert_delta"] = df.apply(
        lambda x: utils.get_days_delta(
            x["ctlog_latest"],
            x["whois_created"]
            if x["whois_created"] and x["ctlog_latest"]
            else None,
        ),
        axis=1,
    )

# set the data type of the domain_to_cert_delta column to float
df["domain_to_latest_cert_delta"] =
df["domain_to_latest_cert_delta"].astype("float64")

# drop columns we imported that we will never use
df = df.drop(columns=["url", "verification_time", "dateadded",
"soa_timestamp","domain_to_cert_delta"])

click.echo(df.head())
click.echo(df.describe(include='all'))
click.echo(df.dtypes)
                            domain  malicious       whois_created
0  i-db5p-cor001.api.p001.1drv.com      False 2013-08-05 18:33:50  \
4       soundcloud-pax.pandora.com      False 1993-12-28 05:00:00
5         joolcomercializadora.com       True 2023-05-22 14:53:50
6        createpdf-asr.acrobat.com      False 1999-03-16 05:00:00
8                          popt.in      False 2016-05-14 16:58:55

       ctlog_earliest        ctlog_latest  ctlog_wildcard
0 2022-01-24 20:01:58 2023-06-08 20:46:06            True  \
4 2022-05-19 00:00:00 2023-06-19 23:59:59            True
5 2022-04-06 22:23:24 2023-09-22 23:59:59           False
6 2022-09-09 00:00:00 2023-10-10 23:59:59            True
8 2023-01-07 20:36:15 2023-08-15 04:16:52           False

   whois_created_dayofweek  ctlog_earliest_dayofweek
ctlog_latest_dayofweek
0                        0                          0
3  \
4                        1                          3
0
5                        0                          2
4
6                        1                          4
1
8                        5                          5
1
```

```
   domain_to_earliest_cert_delta  domain_to_latest_cert_delta
0                        -3095.0                      -3595.0
4                       -10369.0                     -10766.0
5                          410.0                       -124.0
6                        -8578.0                      -8975.0
8                        -2430.0                      -2649.0
                  domain malicious                  whois_created
count              21549     21549                          21549  \
unique             21536         2                            NaN
top     www.mediafire.com    False                            NaN
freq                   2     11739                            NaN
mean                 NaN       NaN  2012-10-03 12:56:32.335050496
min                  NaN       NaN            1986-01-09 00:00:00
25%                  NaN       NaN            2003-05-25 13:35:05
50%                  NaN       NaN            2015-05-07 23:56:05
75%                  NaN       NaN            2023-03-20 15:03:16
max                  NaN       NaN            2023-07-03 08:21:24
std                  NaN       NaN                            NaN

                        ctlog_earliest                   ctlog_latest
count                            21549                          21549  \
unique                             NaN                            NaN
top                                NaN                            NaN
freq                               NaN                            NaN
mean    2022-09-26 15:45:50.943570432  2023-08-14 17:49:06.400900352
min               2021-11-30 05:24:28            2023-01-01 18:42:11
25%               2022-06-24 13:47:12            2023-07-02 08:11:07
50%               2022-10-18 21:00:14            2023-08-21 21:40:11
75%               2022-12-14 00:00:00            2023-09-21 19:41:38
max               2023-06-28 04:36:22            2023-12-31 23:59:59
std                                NaN                            NaN

       ctlog_wildcard  whois_created_dayofweek  ctlog_earliest_dayofweek
count           21549             21549.000000              21549.000000  \
unique              2                      NaN                       NaN
top             False                      NaN                       NaN
freq            13032                      NaN                       NaN
mean              NaN                 2.332823                  2.399462
min               NaN                 0.000000                  0.000000
25%               NaN                 1.000000                  1.000000
50%               NaN                 2.000000                  2.000000
75%               NaN                 4.000000                  4.000000
max               NaN                 6.000000                  6.000000
std               NaN                 1.775043                  1.897252

        ctlog_latest_dayofweek  domain_to_earliest_cert_delta
count             21549.000000                   21549.000000  \
unique                     NaN                            NaN
```

```
top                        NaN                             NaN
freq                       NaN                             NaN
mean                  2.873080                    -3645.602070
min                   0.000000                   -13445.000000
25%                   1.000000                    -7078.000000
50%                   3.000000                    -2637.000000
75%                   5.000000                       69.000000
max                   6.000000                      524.000000
std                   2.057394                     3790.677119

        domain_to_latest_cert_delta
count                  21549.000000
unique                          NaN
top                             NaN
freq                            NaN
mean                   -3967.678222
min                   -13798.000000
25%                    -7421.000000
50%                    -3009.000000
75%                     -144.000000
max                      135.000000
std                     3852.703681
domain                                   string[python]
malicious                                          bool
whois_created                            datetime64[ns]
ctlog_earliest                           datetime64[ns]
ctlog_latest                             datetime64[ns]
ctlog_wildcard                                     bool
whois_created_dayofweek                           int64
ctlog_earliest_dayofweek                          int64
ctlog_latest_dayofweek                            int64
domain_to_earliest_cert_delta                   float64
domain_to_latest_cert_delta                     float64
dtype: object
```

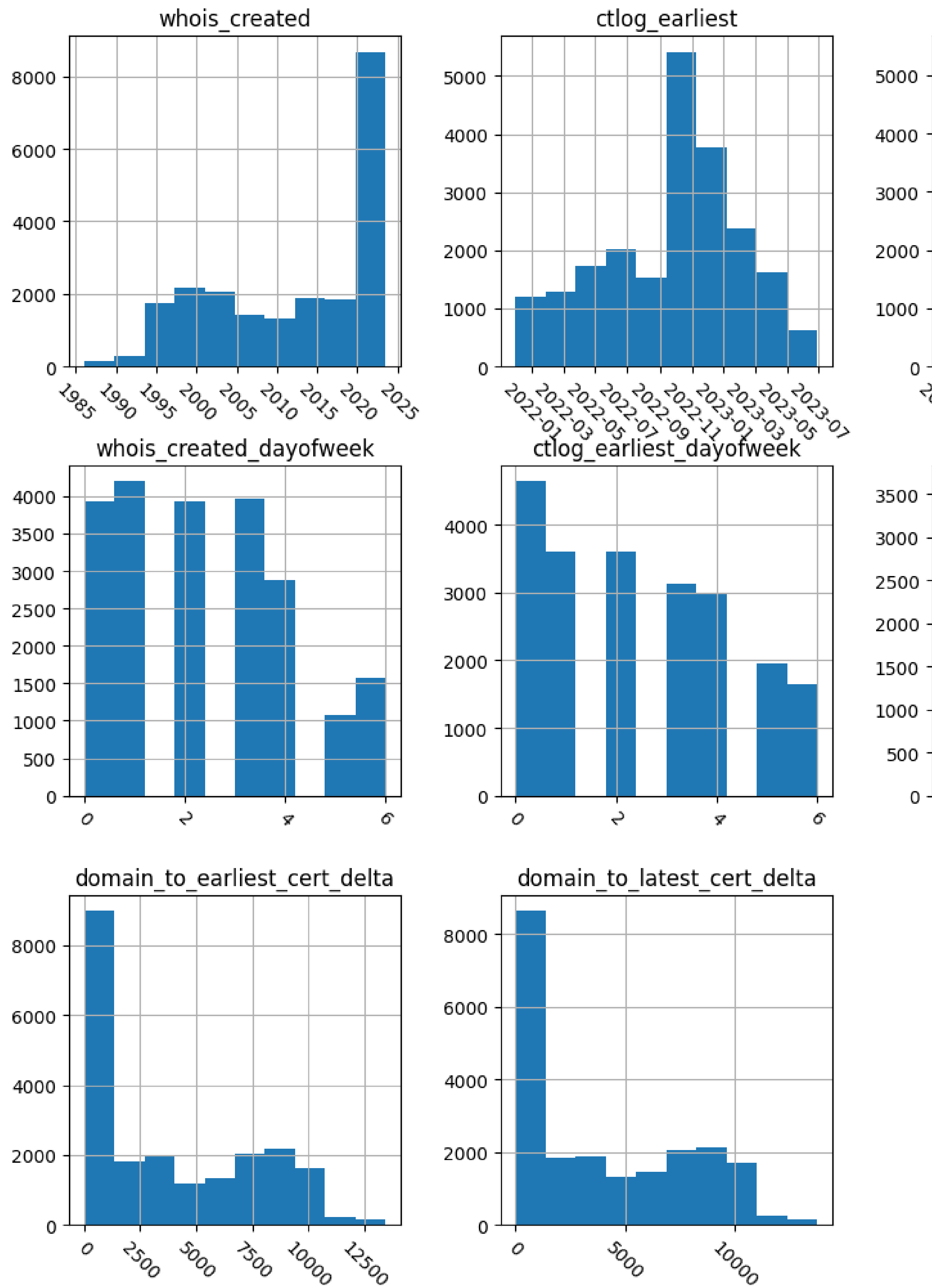
In [3]:

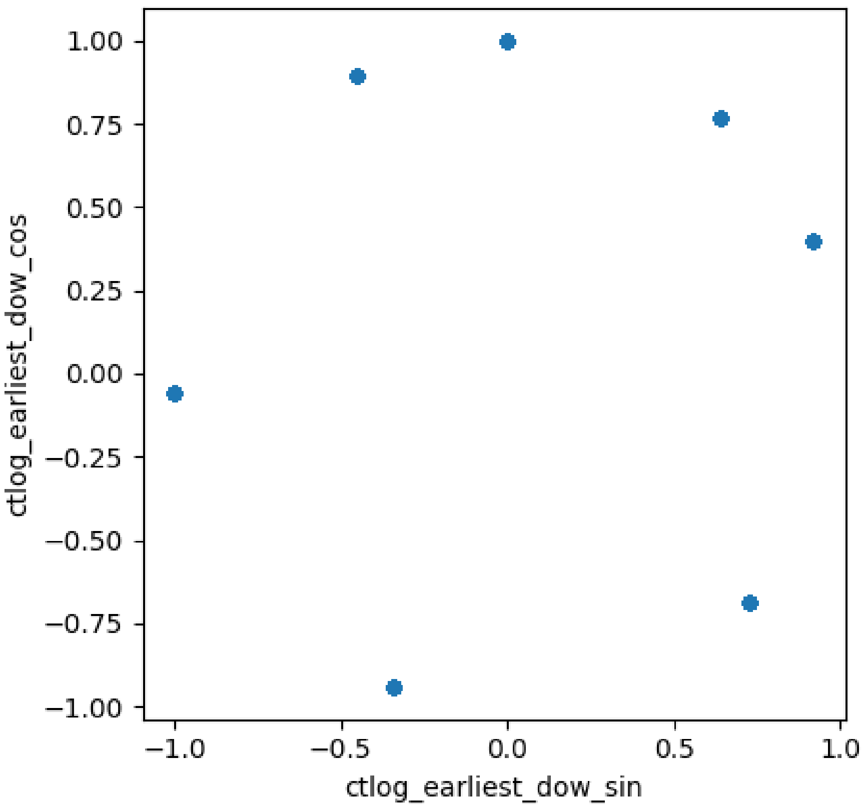
```
# absolute value of the domain_to_cert_delta
df["domain_to_earliest_cert_delta"] = 
abs(df["domain_to_earliest_cert_delta"])
df["domain_to_latest_cert_delta"] = abs(df["domain_to_latest_cert_delta"])

df.hist(figsize=(12,12), xrot=-45)

col_index = df.columns.get_loc("whois_created_dayofweek")
df["whois_created_dow_sin"] = df.apply(lambda row: 
math.sin(360/7*(row[col_index])),axis=1)
df["whois_created_dow_cos"] = df.apply(lambda row: 
math.cos(360/7*(row[col_index])),axis=1)

col_index = df.columns.get_loc("ctlog_earliest_dayofweek")
```

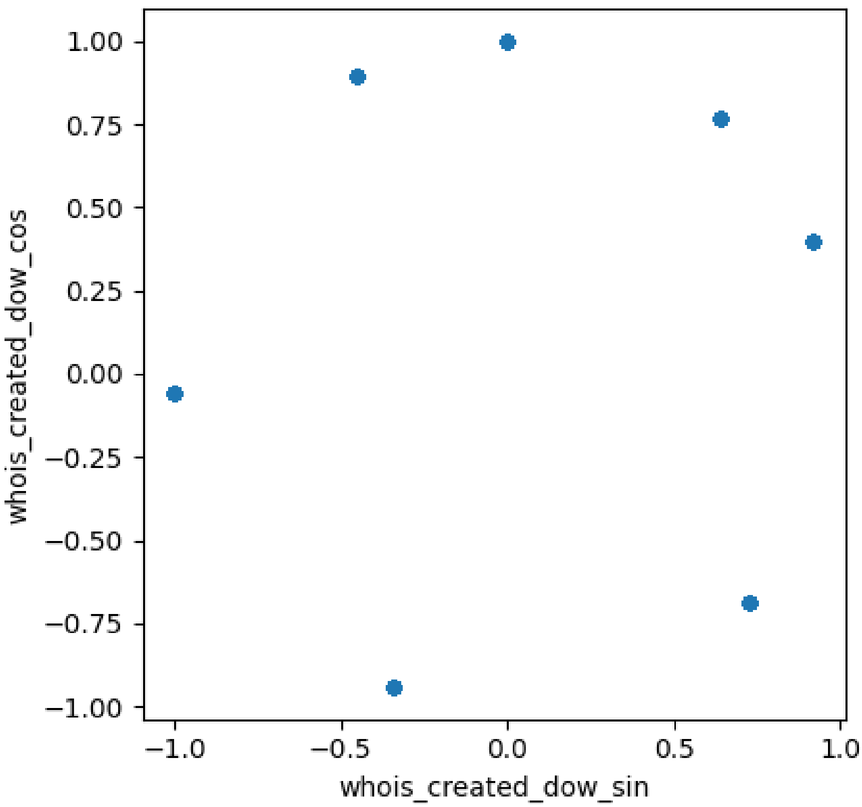
```
df["ctlog_earliest_dow_sin"] = df.apply(lambda row:
math.sin(360/7*(row[col_index])),axis=1)
df["ctlog_earliest_dow_cos"] = df.apply(lambda row:
math.cos(360/7*(row[col_index])),axis=1)

col_index = df.columns.get_loc("ctlog_latest_dayofweek")
df["ctlog_latest_dow_sin"] = df.apply(lambda row:
math.sin(360/7*(row[col_index])),axis=1)
df["ctlog_latest_dow_cos"] = df.apply(lambda row:
math.cos(360/7*(row[col_index])),axis=1)


df.plot.scatter(x="ctlog_earliest_dow_sin",
y="ctlog_earliest_dow_cos").set_aspect('equal')
df.plot.scatter(x="whois_created_dow_sin",
y="whois_created_dow_cos").set_aspect('equal')
df.plot.scatter(x="ctlog_latest_dow_sin",
y="ctlog_latest_dow_cos").set_aspect('equal')

"""
# add one hot encode ctlog_earliest_not_before_dayofweek
df = pd.get_dummies(
        df,
        columns=["ctlog_earliest_dayofweek"],
        prefix=["ctlog_earliest_dow"],
    )
# add one hot encode whois_created_dayofweek to columns
df = pd.get_dummies(
        df, columns=["whois_created_dayofweek"], prefix="whois_dow"
    )
    """

# Summary statistics
click.echo(df.describe(include='all'))
                  domain malicious                  whois_created
count              21549     21549                          21549  \
unique             21536         2                            NaN
top     www.mediafire.com     False                            NaN
freq                   2     11739                            NaN
mean                 NaN       NaN  2012-10-03 12:56:32.335050496
min                  NaN       NaN            1986-01-09 00:00:00
25%                  NaN       NaN            2003-05-25 13:35:05
50%                  NaN       NaN            2015-05-07 23:56:05
75%                  NaN       NaN            2023-03-20 15:03:16
max                  NaN       NaN            2023-07-03 08:21:24
std                  NaN       NaN                            NaN

                      ctlog_earliest                   ctlog_latest
count                          21549                          21549  \
```

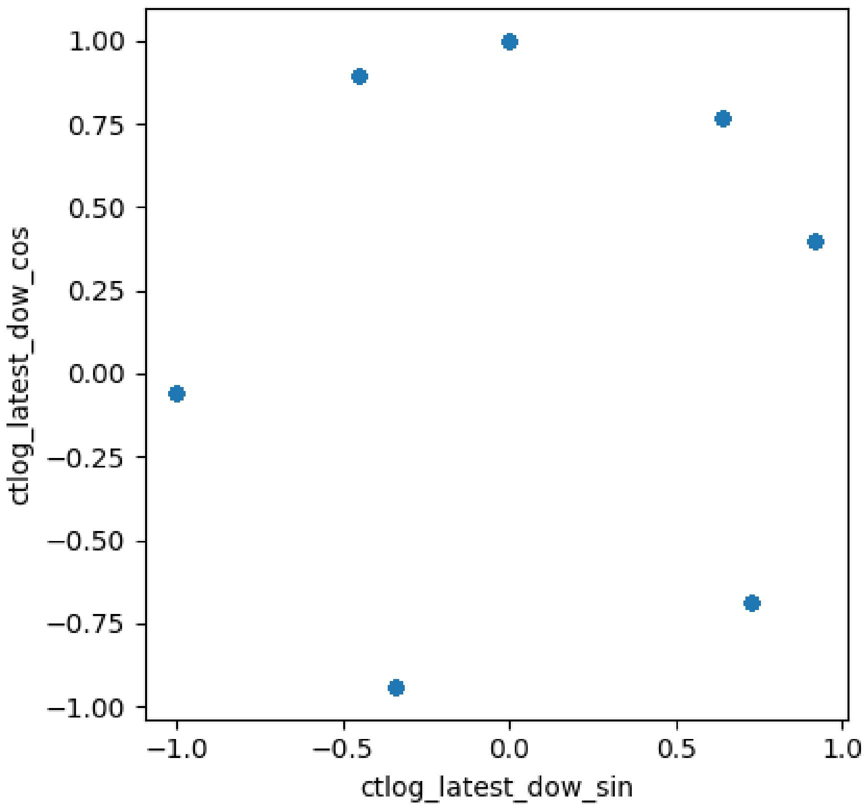
```
unique                         NaN                            NaN
top                            NaN                            NaN
freq                           NaN                            NaN
mean    2022-09-26 15:45:50.943570432  2023-08-14 17:49:06.400900352
min               2021-11-30 05:24:28            2023-01-01 18:42:11
25%               2022-06-24 13:47:12            2023-07-02 08:11:07
50%               2022-10-18 21:00:14            2023-08-21 21:40:11
75%               2022-12-14 00:00:00            2023-09-21 19:41:38
max               2023-06-28 04:36:22            2023-12-31 23:59:59
std                            NaN                            NaN

       ctlog_wildcard  whois_created_dayofweek  ctlog_earliest_dayofweek
count           21549             21549.000000              21549.000000  \
unique              2                      NaN                       NaN
top             False                      NaN                       NaN
freq            13032                      NaN                       NaN
mean              NaN                 2.332823                  2.399462
min               NaN                 0.000000                  0.000000
25%               NaN                 1.000000                  1.000000
50%               NaN                 2.000000                  2.000000
75%               NaN                 4.000000                  4.000000
max               NaN                 6.000000                  6.000000
std               NaN                 1.775043                  1.897252

        ctlog_latest_dayofweek  domain_to_earliest_cert_delta
count             21549.000000                   21549.000000  \
unique                     NaN                            NaN
top                        NaN                            NaN
freq                       NaN                            NaN
mean                  2.873080                    3742.948397
min                   0.000000                       0.000000
25%                   1.000000                     181.000000
50%                   3.000000                    2637.000000
75%                   5.000000                    7078.000000
max                   6.000000                   13445.000000
std                   2.057394                    3694.584062

        domain_to_latest_cert_delta  whois_created_dow_sin
count                  21549.000000           21549.000000  \
unique                          NaN                    NaN
top                             NaN                    NaN
freq                            NaN                    NaN
mean                    3969.491206               0.140419
min                        0.000000              -0.998199
25%                      144.000000              -0.340712
50%                     3009.000000               0.000000
75%                     7421.000000               0.728010
max                    13798.000000               0.918032
std                     3850.835626               0.659922
```

```
        whois_created_dow_cos  ctlog_earliest_dow_sin
ctlog_earliest_dow_cos
count            21549.000000            21549.000000
21549.000000  \
unique                    NaN                     NaN
NaN
top                       NaN                     NaN
NaN
freq                      NaN                     NaN
NaN
mean                 0.054288                0.095357
0.161451
min                 -0.940168               -0.998199               -
0.940168
25%                 -0.685567               -0.340712               -
0.685567
50%                  0.396506                0.000000
0.396506
75%                  0.767830                0.728010
0.892589
max                  1.000000                0.918032
1.000000
std                  0.736128                0.651782
0.734891

        ctlog_latest_dow_sin  ctlog_latest_dow_cos
count           21549.000000          21549.000000
unique                   NaN                   NaN
top                      NaN                   NaN
freq                     NaN                   NaN
mean                0.096253              0.255578
min                -0.998199             -0.940168
25%                -0.450871             -0.685567
50%                 0.000000              0.396506
75%                 0.728010              0.892589
max                 0.918032              1.000000
std                 0.651597              0.707728
```


whois_created
0
2000
4000
6000
8000
1985
1990
1995
2000
2005
2010
2015
2020
2025
ctlog_earliest
0
1000
2000
3000
4000
5000
2022-01
2022-03
2022-05
2022-07
2022-09
2022-11
2023-01
2023-03
2023-05
2023-07
whois_created_dayofweek
0
500
1000
1500
2000
2500
3000
3500
4000
0
2
4
6
ctlog_earliest_dayofweek
0
1000
2000
3000
4000
0
2
4
6
domain_to_earliest_cert_delta
0
2000
4000
6000
8000
0
2500
5000
7500
10000
12500
domain_to_latest_cert_delta
0
2000
4000
6000
8000
0
5000
10000


1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
ctlog_earliest_dow_cos
−1.0
−0.5
0.0
0.5
1.0
ctlog_earliest_dow_sin

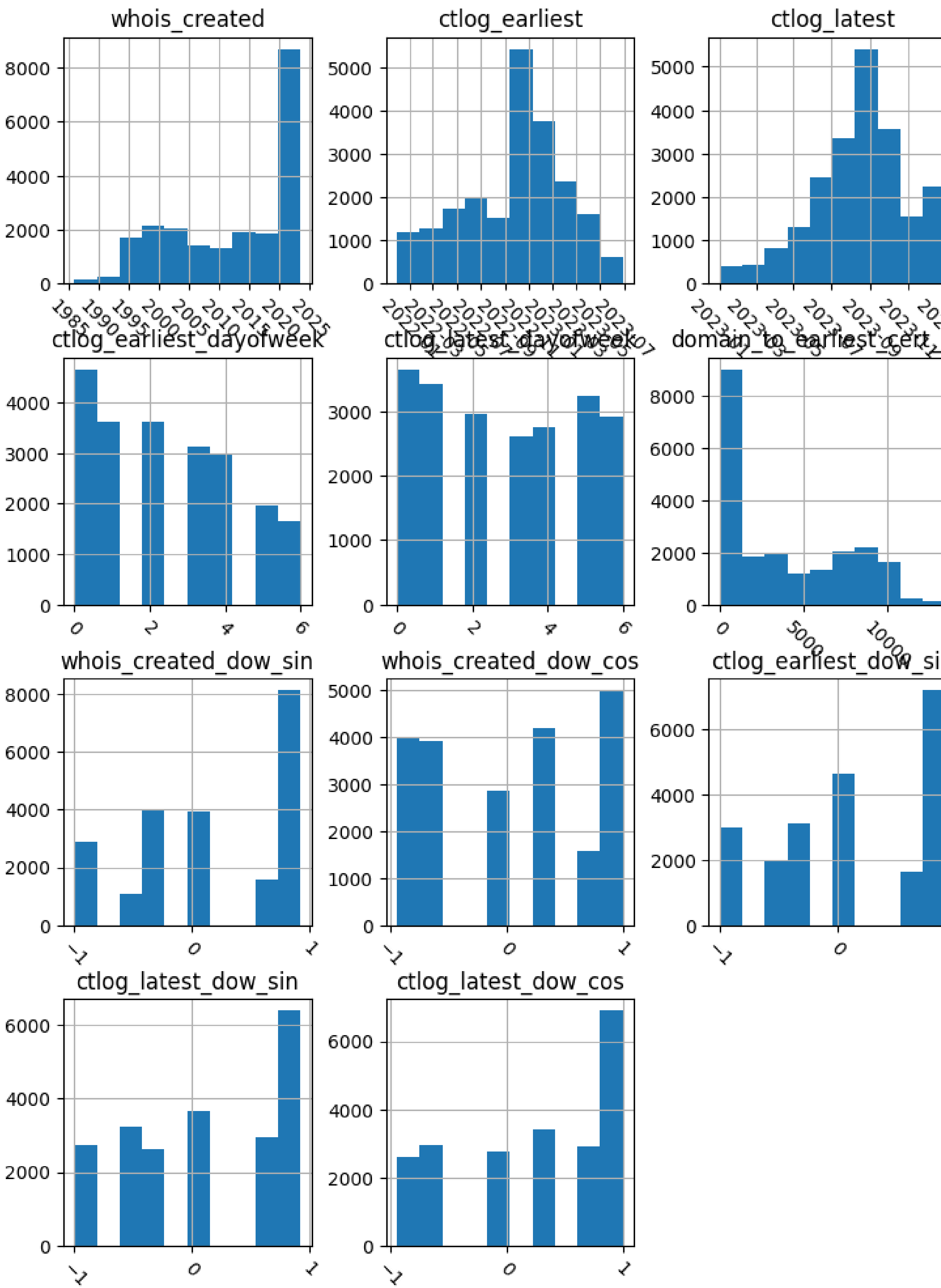

1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
whois_created_dow_cos
−1.0
−0.5
0.0
0.5
1.0
whois_created_dow_sin

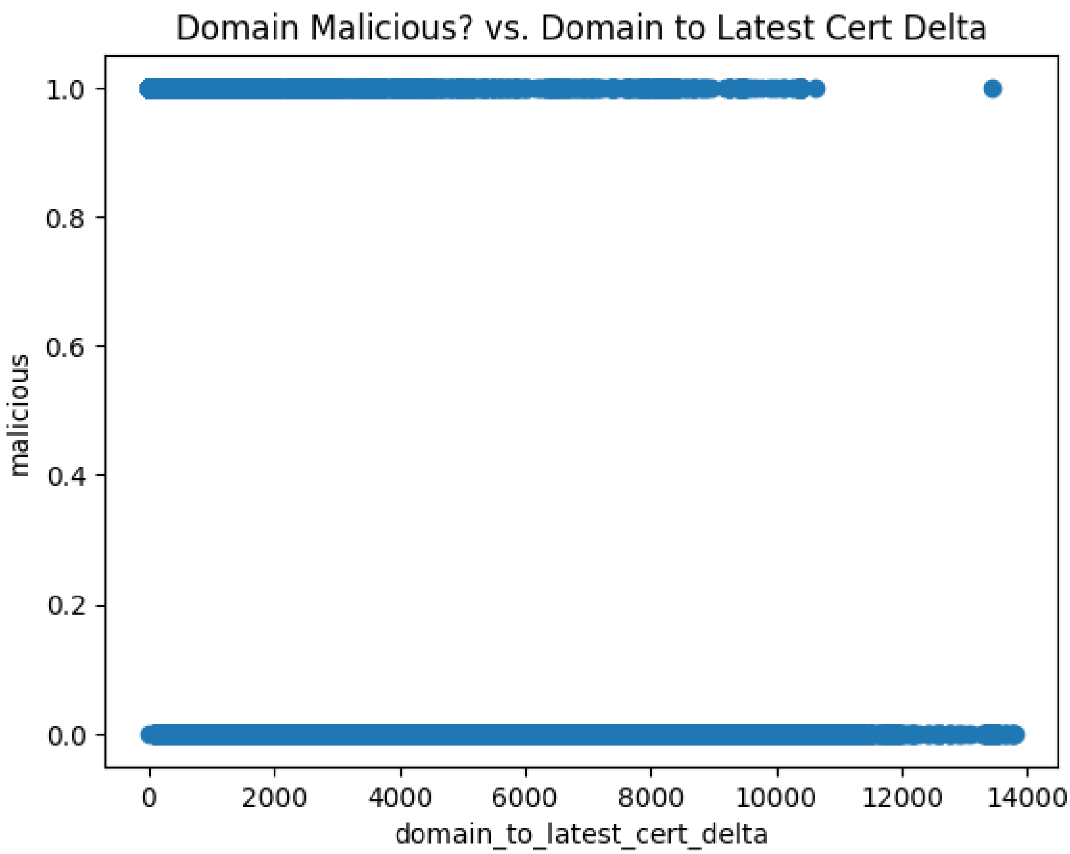


In [4]:

```
# Plot histograms
df.hist(figsize=(12,12), xrot=-45)

# Create a scatter plot of the data, showing malicious and benign domains
# plot the data as a scatter plot
plt.scatter(df["domain_to_earliest_cert_delta"], df["malicious"])
plt.xlabel("domain_to_earliest_cert_delta")
plt.ylabel("malicious")
plt.title("Domain Malicious? vs. Domain to Earliest Cert Delta")
plt.show()
# and for the latest
plt.scatter(df["domain_to_latest_cert_delta"], df["malicious"])
plt.xlabel("domain_to_latest_cert_delta")
plt.ylabel("malicious")
plt.title("Domain Malicious? vs. Domain to Latest Cert Delta")
plt.show()



click.echo(df.head())

# print a count of malicious and benign values
click.echo(df["malicious"].value_counts())
# deduplicate any rows, there shouldn't be any, but just in case
df = df.drop_duplicates()
click.echo(df["malicious"].value_counts())
```

```
click.echo(df.head())

X = df.drop(["malicious","domain", "ctlog_earliest", "ctlog_latest",
"whois_created", "whois_created_dayofweek", "ctlog_earliest_dayofweek"],
axis=1)
X = df.filter(combo_features)
y = df.filter(["malicious"])


click.echo(X.describe())
```


whois_created
ctlog_earliest
ctlog_latest
ctlog_earliest_dayofweek
ctlog_latest_dayofweek
domain_to_earliest_cert_
whois_created_dow_sin
whois_created_dow_cos
ctlog_earliest_dow_si
ctlog_latest_dow_sin
ctlog_latest_dow_cos




```
                                domain  malicious       whois_created
0   i-db5p-cor001.api.p001.1drv.com      False 2013-08-05 18:33:50   \
4        soundcloud-pax.pandora.com      False 1993-12-28 05:00:00
5          joolcomercializadora.com       True 2023-05-22 14:53:50
6         createpdf-asr.acrobat.com      False 1999-03-16 05:00:00
8                           popt.in      False 2016-05-14 16:58:55

        ctlog_earliest         ctlog_latest  ctlog_wildcard
0 2022-01-24 20:01:58 2023-06-08 20:46:06            True   \
4 2022-05-19 00:00:00 2023-06-19 23:59:59            True
5 2022-04-06 22:23:24 2023-09-22 23:59:59           False
6 2022-09-09 00:00:00 2023-10-10 23:59:59            True
8 2023-01-07 20:36:15 2023-08-15 04:16:52           False

   whois_created_dayofweek  ctlog_earliest_dayofweek
ctlog_latest_dayofweek
0                        0                          0
3   \
4                        1                          3
0
5                        0                          2
4
6                        1                          4
1
8                        5                          5
1
```

```
   domain_to_earliest_cert_delta  domain_to_latest_cert_delta
0                         3095.0                       3595.0  \
4                        10369.0                      10766.0
5                          410.0                        124.0
6                         8578.0                       8975.0
8                         2430.0                       2649.0

   whois_created_dow_sin  whois_created_dow_cos  ctlog_earliest_dow_sin
0               0.000000               1.000000               0.000000  \
4               0.918032               0.396506              -0.340712
5               0.000000               1.000000               0.728010
6               0.918032               0.396506              -0.998199
8              -0.450871               0.892589              -0.450871

   ctlog_earliest_dow_cos  ctlog_latest_dow_sin  ctlog_latest_dow_cos
0                1.000000             -0.340712             -0.940168
4               -0.940168              0.000000              1.000000
5               -0.685567             -0.998199             -0.059997
6               -0.059997              0.918032              0.396506
8                0.892589              0.918032              0.396506
malicious
False    11739
True      9810
Name: count, dtype: int64
malicious
False    11739
True      9810
Name: count, dtype: int64
                           domain  malicious       whois_created
0  i-db5p-cor001.api.p001.1drv.com      False 2013-08-05 18:33:50  \
4       soundcloud-pax.pandora.com      False 1993-12-28 05:00:00
5         joolcomercializadora.com       True 2023-05-22 14:53:50
6        createpdf-asr.acrobat.com      False 1999-03-16 05:00:00
8                          popt.in      False 2016-05-14 16:58:55

       ctlog_earliest        ctlog_latest  ctlog_wildcard
0 2022-01-24 20:01:58 2023-06-08 20:46:06            True  \
4 2022-05-19 00:00:00 2023-06-19 23:59:59            True
5 2022-04-06 22:23:24 2023-09-22 23:59:59           False
6 2022-09-09 00:00:00 2023-10-10 23:59:59            True
8 2023-01-07 20:36:15 2023-08-15 04:16:52           False

   whois_created_dayofweek  ctlog_earliest_dayofweek
ctlog_latest_dayofweek
0                        0                         0
3  \
4                        1                         3
0
```

```
5                       0                         2
4
6                       1                         4
1
8                       5                         5
1

   domain_to_earliest_cert_delta  domain_to_latest_cert_delta
0                         3095.0                       3595.0  \
4                        10369.0                      10766.0
5                          410.0                        124.0
6                         8578.0                       8975.0
8                         2430.0                       2649.0

   whois_created_dow_sin  whois_created_dow_cos  ctlog_earliest_dow_sin
0               0.000000               1.000000                0.000000  \
4               0.918032               0.396506               -0.340712
5               0.000000               1.000000                0.728010
6               0.918032               0.396506               -0.998199
8              -0.450871               0.892589               -0.450871

   ctlog_earliest_dow_cos  ctlog_latest_dow_sin  ctlog_latest_dow_cos
0                1.000000             -0.340712             -0.940168
4               -0.940168              0.000000              1.000000
5               -0.685567             -0.998199             -0.059997
6               -0.059997              0.918032              0.396506
8                0.892589              0.918032              0.396506
       domain_to_earliest_cert_delta
count                   21549.000000
mean                     3742.948397
std                      3694.584062
min                         0.000000
25%                       181.000000
50%                      2637.000000
75%                      7078.000000
max                     13445.000000
```

In [5]:

```
# convert y (malicious) to 1/0 int
y = y.astype('int')
# convert X["ctlog_wildcard"]  to 1/0 int
if "ctlog_wildcard" in X.columns:
    X["ctlog_wildcard"] = X["ctlog_wildcard"].astype('int')

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

X_train = sm.add_constant(X_train)
X_test = sm.add_constant(X_test)
```

```
smfit = sm.Logit(y_train,X_train).fit()

smfit.summary()
Optimization terminated successfully.
         Current function value: 0.355908
         Iterations 7
```

Out[5]:

Logit Regression Results

Dep. Variable:	malicious	**No. Observations:**	17239
Model:	Logit	**Df Residuals:**	17236
Method:	MLE	**Df Model:**	2
Date:	Tue, 08 Aug 2023	**Pseudo R-squ.:**	0.4838
Time:	19:09:25	**Log-Likelihood:**	-6135.5
converged:	True	**LL-Null:**	-11885.
Covariance Type:	nonrobust	**LLR p-value:**	0.000

	coef	**std err**	**z**	**P>|z|**	**[0.025**	**0.975]**
const	2.0751	0.035	60.093	0.000	2.007	2.143
domain_to_earliest_cert_delta	-0.0006	9.83e-06	-61.683	0.000	-0.001	-0.001
ctlog_wildcard	-1.1842	0.049	-24.393	0.000	-1.279	-1.089

In [6]:

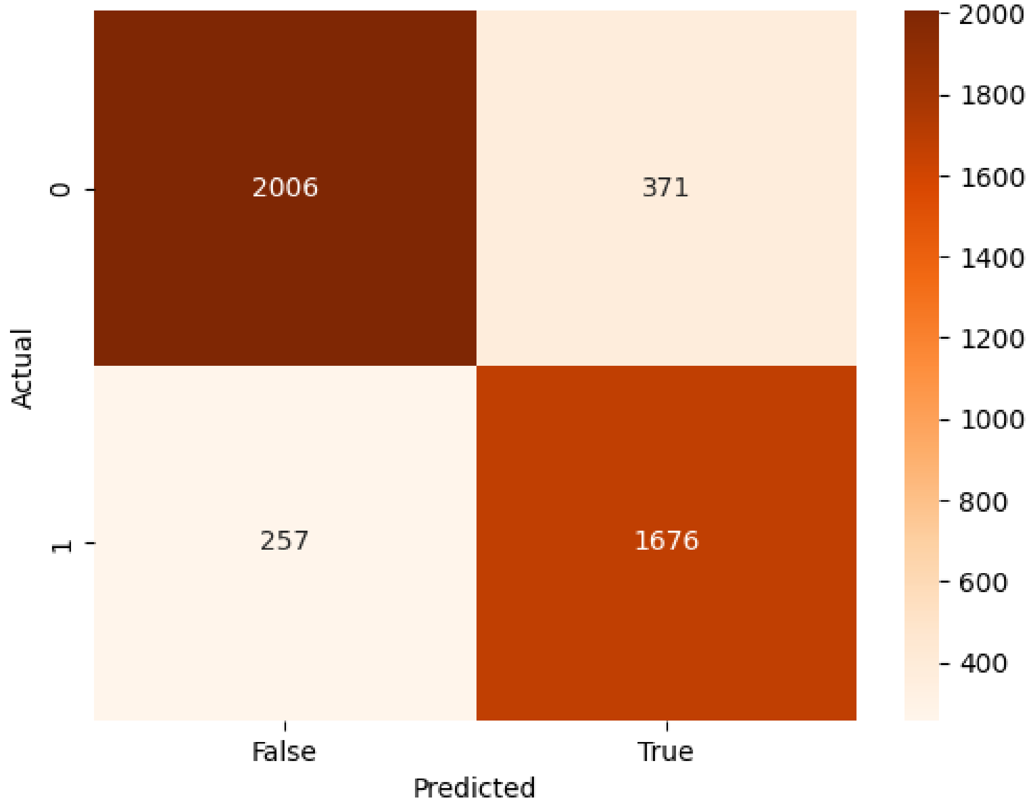
```
# Predict the malicious column using the test data
#add the incepts


y_predicted = smfit.predict(X_test)

# Present the results in a confusion matrix
confusion_matrix = confusion_matrix(y_test, y_predicted.round())
click.echo(confusion_matrix)

click.echo("Classification report:")
click.echo(classification_report(y_test, y_predicted.round()))


# Heatmap of confusion matrix
y_predicted

threshold = 0.5
y_predicted = [y > threshold for y in y_predicted]
data = {'Actual':    y_test.values.flatten(),
        'Predicted': y_predicted
        }

# Generate a confusion matrix and heatmap to evaluate the Type I and Type
II errors/ FP/FN etc.
df = pd.DataFrame(data, columns=['Actual','Predicted'])
confusion_matrix = pd.crosstab(df['Actual'], df['Predicted'],
rownames=['Actual'], colnames=['Predicted'])
```

```
fig = sns.heatmap(confusion_matrix, annot=True, cmap='Oranges', fmt='g')
fig
```

```
[[2006  371]
 [ 257 1676]]
Classification report:
              precision    recall  f1-score   support

           0       0.89      0.84      0.86      2377
           1       0.82      0.87      0.84      1933

    accuracy                           0.85      4310
   macro avg       0.85      0.86      0.85      4310
weighted avg       0.86      0.85      0.85      4310
```

Out[6]:

```
<Axes: xlabel='Predicted', ylabel='Actual'>
```



IV. Feature Set C

In [1]:

```
import click
import pandas as pd
import glob
import os
#import sklearn
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report
from sklearn.preprocessing import StandardScaler
from scipy import stats
from sklearn.linear_model import LogisticRegression
import statsmodels.api as sm
import matplotlib.pyplot as plt
from datetime import datetime, date
# qqplot of the data
import seaborn as sns
import math
import numpy as np
import utils

combo_features = [
    'domain_to_earliest_cert_delta',
    'ctlog_earliest_dow_sin',
    'ctlog_earliest_dow_cos',
    'ctlog_wildcard'
    ]


path = "../data/"
filename = None

epoch = datetime(1970,1,1)
# open the training data file, from ../data/ for the most recent merged
training data file saved by prepare-training-data-parallel.py
full_path = path + "merged_20230705-104357_training_adorned-engineered.csv"
# get latest file matching the glob
if not filename:
    filename = max(glob.glob(full_path), key=os.path.getctime)
    click.echo(f"Using {filename} as training data")
    df = pd.read_csv(filename, sep=",")

# shuffle the rows
df = df.sample(frac=1, random_state=42).reset_index(drop=True)

click.echo(df.shape)
click.echo(df.columns)
```

```
""" # From prepare-training-data-parallel.py:
# Columns:
url
verification_time
domain
malicious
dateadded
whois_created
soa_timestamp
ctlog_earliest
ctlog_latest
ctlog_wildcard
whois_created_dayofweek,
ctlog_earliest_dayofweek,
domain_to_cert_delta
"""


# Data cleaning - there may be some epoch values in the whois_created
#  & ct_log_earliest columns, so we need to remove those rows

# convert the whois_created and ctlog_earliest, ctlog_latest columns to
datetime

df = df.loc[df["whois_created"] != ""]
df = df.loc[df["ctlog_earliest"] != ""]
df = df.loc[df["ctlog_latest"] != ""]

# some dates are have nanoseconds in them, so we need to remove the
nanosecond part
df["whois_created"] = df["whois_created"].str.split(".", n = 1, expand =
True)[0]
# some dates has additional timezone information, so we need to remove that
df["whois_created"] = df["whois_created"].apply(lambda x: " ".join(
str(x).split(" ")[:2]))


# convert the whois_created and ct_log_earliest columns to datetime
df = df.astype({"whois_created": 'datetime64[ns]', "ctlog_earliest":
'datetime64[ns]', "ctlog_latest": 'datetime64[ns]'})
df = df.loc[df["whois_created"].ne(pd.Timestamp('1970-01-01 00:00:00'))]
df = df.loc[df["ctlog_earliest"].ne(pd.Timestamp('1970-01-01 00:00:00'))]
df = df.loc[df["ctlog_latest"].ne(pd.Timestamp('1970-01-01 00:00:00'))]


# malicious is a bool
df['malicious'] = df['malicious'].astype('bool')
df['domain'] = df['domain'].astype('string')
```

```
Using ../data/merged_20230705-104357_training_adorned-engineered.csv as
training data
(35438, 13)
Index(['url', 'verification_time', 'domain', 'malicious', 'dateadded',
       'whois_created', 'soa_timestamp', 'ctlog_earliest', 'ctlog_latest',
       'ctlog_wildcard', 'whois_created_dayofweek',
'ctlog_earliest_dayofweek',
       'domain_to_cert_delta'],
      dtype='object')
```

In [2]:

```
#########
# Feature engineering
#########

# remove any rows where the whois_created or ctlog_earliest columns are
null
df = df.loc[df["whois_created"].notnull()]
df = df.loc[df["ctlog_earliest"].notnull()]

# if the data is missing the "whois_created_dayofweek" or
"ctlog_earliest_dayofweek" columns, then add them
# whois created day of the week 0 = Monday, 6 = Sunday
df["whois_created_dayofweek"] = df["whois_created"].apply(
        lambda x: x.weekday() if isinstance(x, date) else None
    )

# cert valid day of the week 0 = Monday, 6 = Sunday
df["ctlog_earliest_dayofweek"] = df["ctlog_earliest"].apply(
        lambda x: x.weekday() if isinstance(x, date) else None
    )

df["ctlog_latest_dayofweek"] = df["ctlog_latest"].apply(
        lambda x: x.weekday() if isinstance(x, date) else None
    )

# set data type of the day of week columns to int
df["whois_created_dayofweek"] =
df["whois_created_dayofweek"].astype("int64")
df["ctlog_earliest_dayofweek"] =
df["ctlog_earliest_dayofweek"].astype("int64")
df["ctlog_latest_dayofweek"] = df["ctlog_latest_dayofweek"].astype("int64")

# domain to cert delta
df["domain_to_earliest_cert_delta"] = df.apply(
        lambda x: utils.get_days_delta(
            x["ctlog_earliest"],
            x["whois_created"]
            if x["whois_created"] and x["ctlog_earliest"]
            else None,
```

```
        ),
        axis=1,
    )

# set the data type of the domain_to_cert_delta column to float
df["domain_to_earliest_cert_delta"] =
df["domain_to_earliest_cert_delta"].astype("float64")

# domain to latest cert delta
df["domain_to_latest_cert_delta"] = df.apply(
        lambda x: utils.get_days_delta(
            x["ctlog_latest"],
            x["whois_created"]
            if x["whois_created"] and x["ctlog_latest"]
            else None,
        ),
        axis=1,
    )

# set the data type of the domain_to_cert_delta column to float
df["domain_to_latest_cert_delta"] =
df["domain_to_latest_cert_delta"].astype("float64")

# drop columns we imported that we will never use
df = df.drop(columns=["url", "verification_time", "dateadded",
"soa_timestamp","domain_to_cert_delta"])

click.echo(df.head())
click.echo(df.describe(include='all'))
click.echo(df.dtypes)
                           domain  malicious       whois_created
0  i-db5p-cor001.api.p001.1drv.com      False 2013-08-05 18:33:50  \
4       soundcloud-pax.pandora.com      False 1993-12-28 05:00:00
5         joolcomercializadora.com       True 2023-05-22 14:53:50
6        createpdf-asr.acrobat.com      False 1999-03-16 05:00:00
8                          popt.in      False 2016-05-14 16:58:55

       ctlog_earliest        ctlog_latest  ctlog_wildcard
0 2022-01-24 20:01:58 2023-06-08 20:46:06            True  \
4 2022-05-19 00:00:00 2023-06-19 23:59:59            True
5 2022-04-06 22:23:24 2023-09-22 23:59:59           False
6 2022-09-09 00:00:00 2023-10-10 23:59:59            True
8 2023-01-07 20:36:15 2023-08-15 04:16:52           False

   whois_created_dayofweek  ctlog_earliest_dayofweek
ctlog_latest_dayofweek
0                        0                          0
3  \
```

```
4                        1                        3
0
5                        0                        2
4
6                        1                        4
1
8                        5                        5
1

   domain_to_earliest_cert_delta  domain_to_latest_cert_delta
0                        -3095.0                      -3595.0
4                       -10369.0                     -10766.0
5                          410.0                       -124.0
6                        -8578.0                      -8975.0
8                        -2430.0                      -2649.0
                   domain malicious                  whois_created
count               21549     21549                          21549  \
unique              21536         2                            NaN
top     www.mediafire.com     False                            NaN
freq                    2     11739                            NaN
mean                  NaN       NaN  2012-10-03 12:56:32.335050496
min                   NaN       NaN            1986-01-09 00:00:00
25%                   NaN       NaN            2003-05-25 13:35:05
50%                   NaN       NaN            2015-05-07 23:56:05
75%                   NaN       NaN            2023-03-20 15:03:16
max                   NaN       NaN            2023-07-03 08:21:24
std                   NaN       NaN                            NaN

                       ctlog_earliest                   ctlog_latest
count                           21549                          21549  \
unique                            NaN                            NaN
top                               NaN                            NaN
freq                              NaN                            NaN
mean    2022-09-26 15:45:50.943570432  2023-08-14 17:49:06.400900352
min               2021-11-30 05:24:28            2023-01-01 18:42:11
25%               2022-06-24 13:47:12            2023-07-02 08:11:07
50%               2022-10-18 21:00:14            2023-08-21 21:40:11
75%               2022-12-14 00:00:00            2023-09-21 19:41:38
max               2023-06-28 04:36:22            2023-12-31 23:59:59
std                               NaN                            NaN

       ctlog_wildcard  whois_created_dayofweek  ctlog_earliest_dayofweek
count           21549             21549.000000              21549.000000  \
unique              2                      NaN                       NaN
top             False                      NaN                       NaN
freq            13032                      NaN                       NaN
mean              NaN                 2.332823                  2.399462
min               NaN                 0.000000                  0.000000
25%               NaN                 1.000000                  1.000000
```

```
50%              NaN                   2.000000                   2.000000
75%              NaN                   4.000000                   4.000000
max              NaN                   6.000000                   6.000000
std              NaN                   1.775043                   1.897252

        ctlog_latest_dayofweek  domain_to_earliest_cert_delta
count            21549.000000                   21549.000000  \
unique                    NaN                            NaN
top                       NaN                            NaN
freq                      NaN                            NaN
mean                 2.873080                   -3645.602070
min                  0.000000                  -13445.000000
25%                  1.000000                   -7078.000000
50%                  3.000000                   -2637.000000
75%                  5.000000                      69.000000
max                  6.000000                     524.000000
std                  2.057394                    3790.677119

        domain_to_latest_cert_delta
count                  21549.000000
unique                          NaN
top                             NaN
freq                            NaN
mean                   -3967.678222
min                   -13798.000000
25%                    -7421.000000
50%                    -3009.000000
75%                     -144.000000
max                      135.000000
std                     3852.703681
domain                              string[python]
malicious                                     bool
whois_created                       datetime64[ns]
ctlog_earliest                      datetime64[ns]
ctlog_latest                        datetime64[ns]
ctlog_wildcard                                bool
whois_created_dayofweek                      int64
ctlog_earliest_dayofweek                     int64
ctlog_latest_dayofweek                       int64
domain_to_earliest_cert_delta              float64
domain_to_latest_cert_delta                float64
dtype: object
```

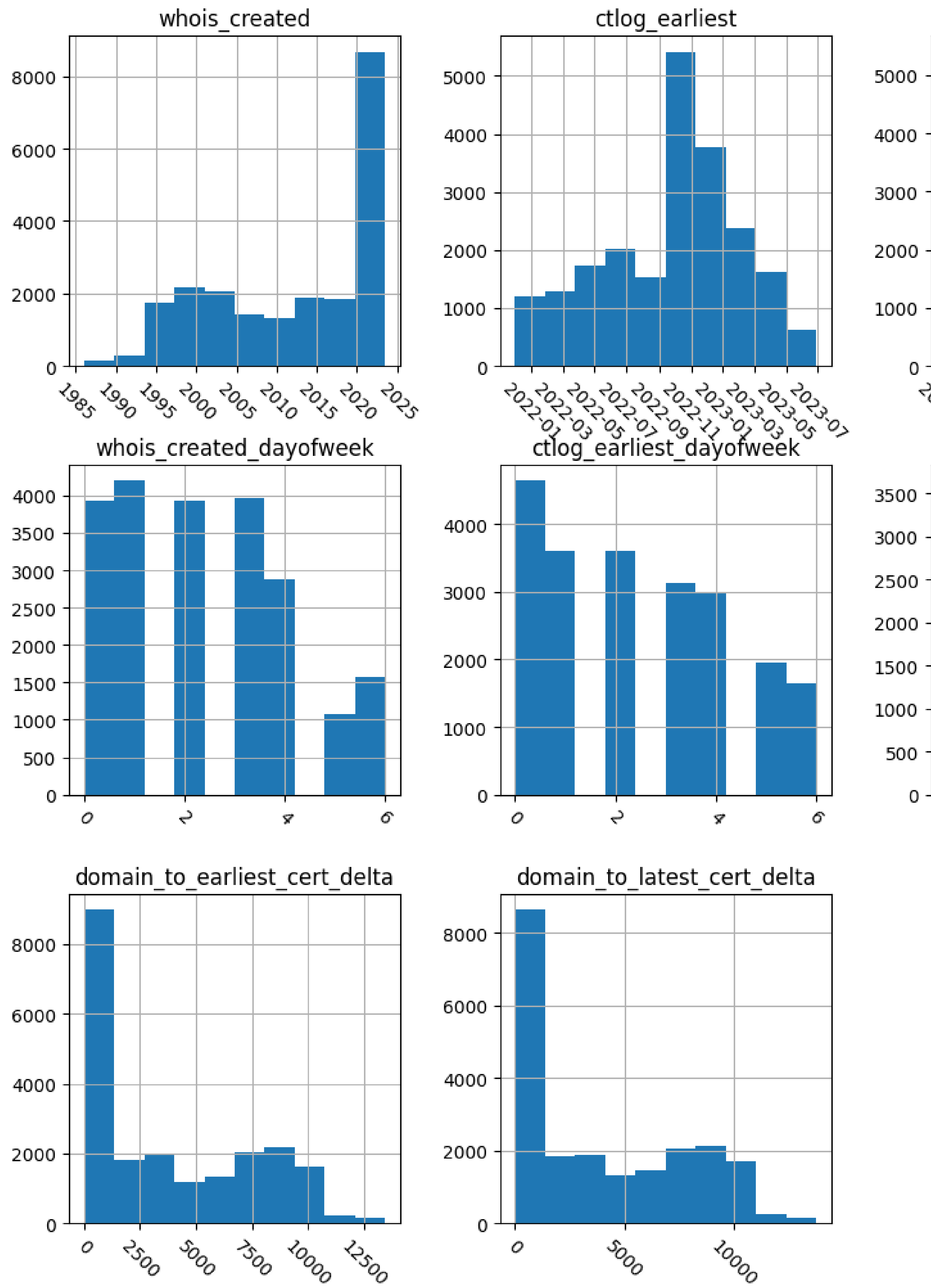
In [3]:

```
# absolute value of the domain_to_cert_delta
df["domain_to_earliest_cert_delta"] =
abs(df["domain_to_earliest_cert_delta"])
df["domain_to_latest_cert_delta"] = abs(df["domain_to_latest_cert_delta"])

df.hist(figsize=(12,12), xrot=-45)
```

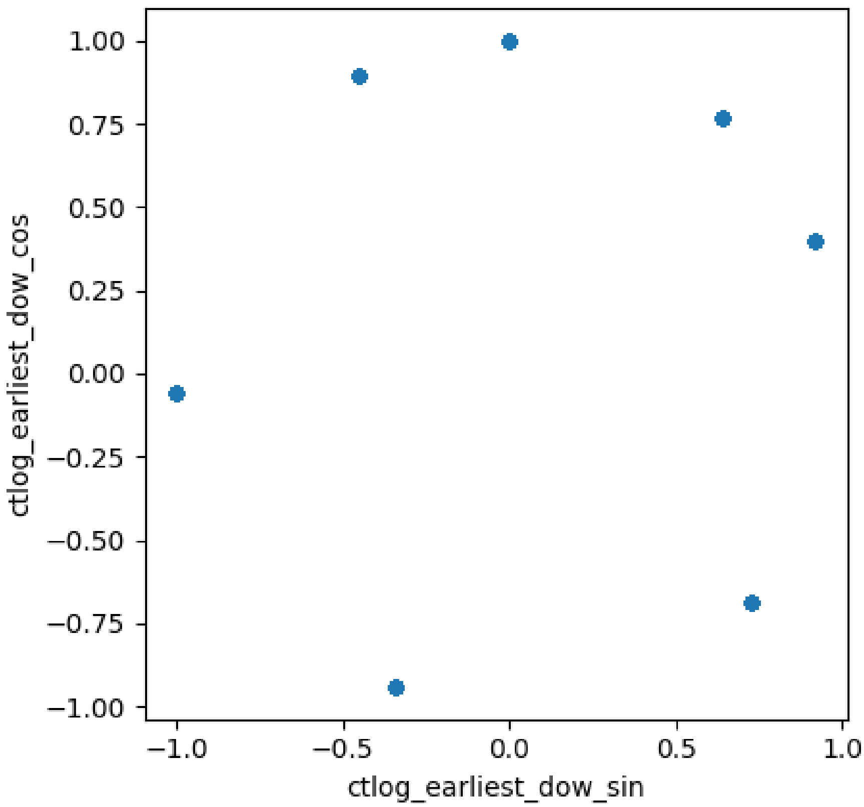
```
col_index = df.columns.get_loc("whois_created_dayofweek")
df["whois_created_dow_sin"] = df.apply(lambda row:
math.sin(360/7*(row[col_index])),axis=1)
df["whois_created_dow_cos"] = df.apply(lambda row:
math.cos(360/7*(row[col_index])),axis=1)

col_index = df.columns.get_loc("ctlog_earliest_dayofweek")
df["ctlog_earliest_dow_sin"] = df.apply(lambda row:
math.sin(360/7*(row[col_index])),axis=1)
df["ctlog_earliest_dow_cos"] = df.apply(lambda row:
math.cos(360/7*(row[col_index])),axis=1)

col_index = df.columns.get_loc("ctlog_latest_dayofweek")
df["ctlog_latest_dow_sin"] = df.apply(lambda row:
math.sin(360/7*(row[col_index])),axis=1)
df["ctlog_latest_dow_cos"] = df.apply(lambda row:
math.cos(360/7*(row[col_index])),axis=1)


df.plot.scatter(x="ctlog_earliest_dow_sin",
y="ctlog_earliest_dow_cos").set_aspect('equal')
df.plot.scatter(x="whois_created_dow_sin",
y="whois_created_dow_cos").set_aspect('equal')
df.plot.scatter(x="ctlog_latest_dow_sin",
y="ctlog_latest_dow_cos").set_aspect('equal')

"""
# add one hot encode ctlog_earliest_not_before_dayofweek
df = pd.get_dummies(
        df,
        columns=["ctlog_earliest_dayofweek"],
        prefix=["ctlog_earliest_dow"],
    )
# add one hot encode whois_created_dayofweek to columns
df = pd.get_dummies(
        df, columns=["whois_created_dayofweek"], prefix="whois_dow"
    )
    """

# Summary statistics
click.echo(df.describe(include='all'))
                  domain malicious                  whois_created
count              21549     21549                          21549  \
unique             21536         2                            NaN
top     www.mediafire.com     False                            NaN
freq                   2     11739                            NaN
mean                 NaN       NaN  2012-10-03 12:56:32.335050496
min                  NaN       NaN            1986-01-09 00:00:00
```

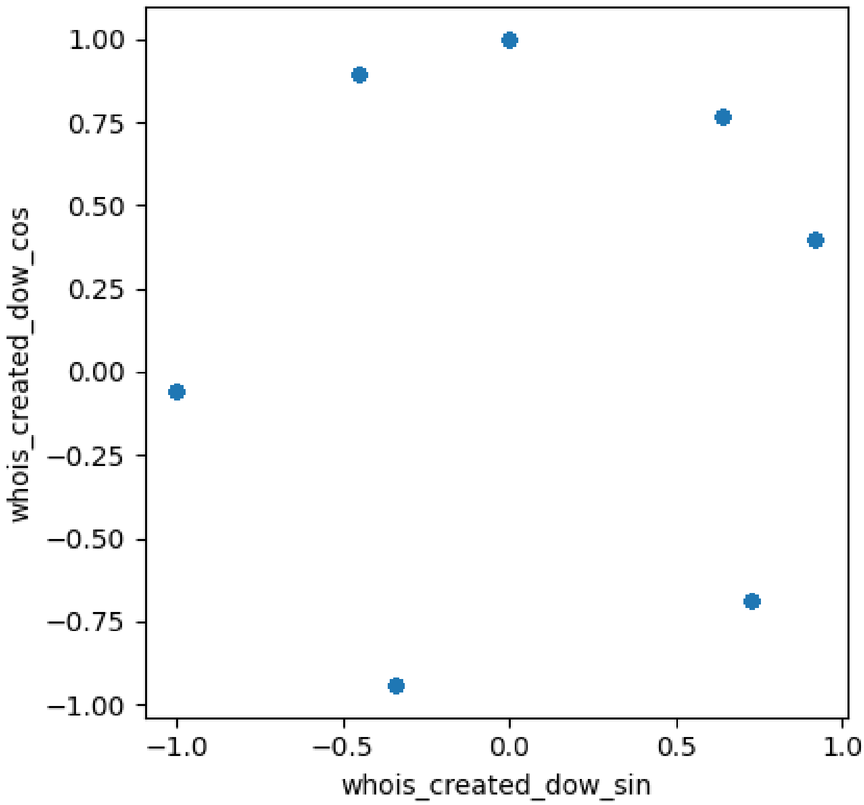
```
25%                   NaN       NaN           2003-05-25 13:35:05
50%                   NaN       NaN           2015-05-07 23:56:05
75%                   NaN       NaN           2023-03-20 15:03:16
max                   NaN       NaN           2023-07-03 08:21:24
std                   NaN       NaN                           NaN

                       ctlog_earliest                  ctlog_latest
count                           21549                         21549  \
unique                            NaN                           NaN
top                               NaN                           NaN
freq                              NaN                           NaN
mean    2022-09-26 15:45:50.943570432 2023-08-14 17:49:06.400900352
min               2021-11-30 05:24:28           2023-01-01 18:42:11
25%               2022-06-24 13:47:12           2023-07-02 08:11:07
50%               2022-10-18 21:00:14           2023-08-21 21:40:11
75%               2022-12-14 00:00:00           2023-09-21 19:41:38
max               2023-06-28 04:36:22           2023-12-31 23:59:59
std                               NaN                           NaN

       ctlog_wildcard  whois_created_dayofweek  ctlog_earliest_dayofweek
count           21549             21549.000000              21549.000000  \
unique              2                      NaN                       NaN
top             False                      NaN                       NaN
freq            13032                      NaN                       NaN
mean              NaN                 2.332823                  2.399462
min               NaN                 0.000000                  0.000000
25%               NaN                 1.000000                  1.000000
50%               NaN                 2.000000                  2.000000
75%               NaN                 4.000000                  4.000000
max               NaN                 6.000000                  6.000000
std               NaN                 1.775043                  1.897252

        ctlog_latest_dayofweek  domain_to_earliest_cert_delta
count             21549.000000                   21549.000000  \
unique                     NaN                            NaN
top                        NaN                            NaN
freq                       NaN                            NaN
mean                  2.873080                    3742.948397
min                   0.000000                       0.000000
25%                   1.000000                     181.000000
50%                   3.000000                    2637.000000
75%                   5.000000                    7078.000000
max                   6.000000                   13445.000000
std                   2.057394                    3694.584062

        domain_to_latest_cert_delta  whois_created_dow_sin
count                  21549.000000           21549.000000  \
unique                          NaN                    NaN
top                             NaN                    NaN
```

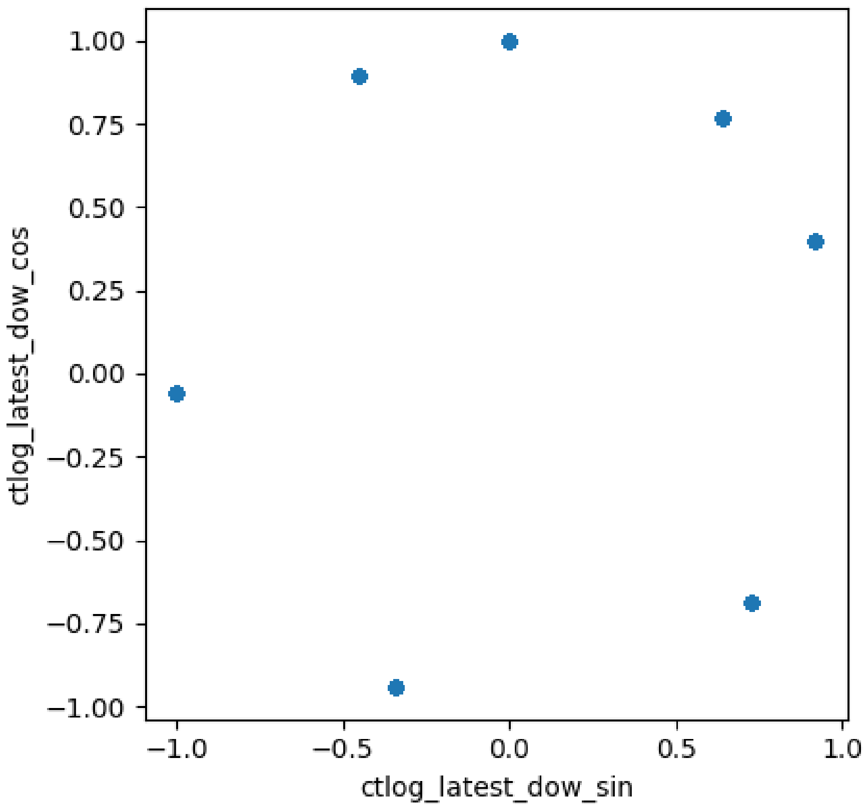
```
freq                          NaN                     NaN
mean                  3969.491206                0.140419
min                      0.000000               -0.998199
25%                    144.000000               -0.340712
50%                   3009.000000                0.000000
75%                   7421.000000                0.728010
max                  13798.000000                0.918032
std                   3850.835626                0.659922

        whois_created_dow_cos  ctlog_earliest_dow_sin
ctlog_earliest_dow_cos
count             21549.000000            21549.000000
21549.000000  \
unique                     NaN                     NaN
NaN
top                        NaN                     NaN
NaN
freq                       NaN                     NaN
NaN
mean                  0.054288                0.095357
0.161451
min                  -0.940168               -0.998199               -
0.940168
25%                  -0.685567               -0.340712               -
0.685567
50%                   0.396506                0.000000
0.396506
75%                   0.767830                0.728010
0.892589
max                   1.000000                0.918032
1.000000
std                   0.736128                0.651782
0.734891

        ctlog_latest_dow_sin  ctlog_latest_dow_cos
count            21549.000000          21549.000000
unique                    NaN                   NaN
top                       NaN                   NaN
freq                      NaN                   NaN
mean                 0.096253              0.255578
min                 -0.998199             -0.940168
25%                 -0.450871             -0.685567
50%                  0.000000              0.396506
75%                  0.728010              0.892589
max                  0.918032              1.000000
std                  0.651597              0.707728
```



whois_created
0
2000
4000
6000
8000
1985
1990
1995
2000
2005
2010
2015
2020
2025
ctlog_earliest
0
1000
2000
3000
4000
5000
2022-01
2022-03
2022-05
2022-07
2022-09
2022-11
2023-01
2023-03
2023-05
2023-07
whois_created_dayofweek
0
500
1000
1500
2000
2500
3000
3500
4000
0
2
4
6
ctlog_earliest_dayofweek
0
1000
2000
3000
4000
0
2
4
6
domain_to_earliest_cert_delta
0
2000
4000
6000
8000
0
2500
5000
7500
10000
12500
domain_to_latest_cert_delta
0
2000
4000
6000
8000
0
5000
10000


1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
ctlog_earliest_dow_cos
−1.0
−0.5
0.0
0.5
1.0
ctlog_earliest_dow_sin

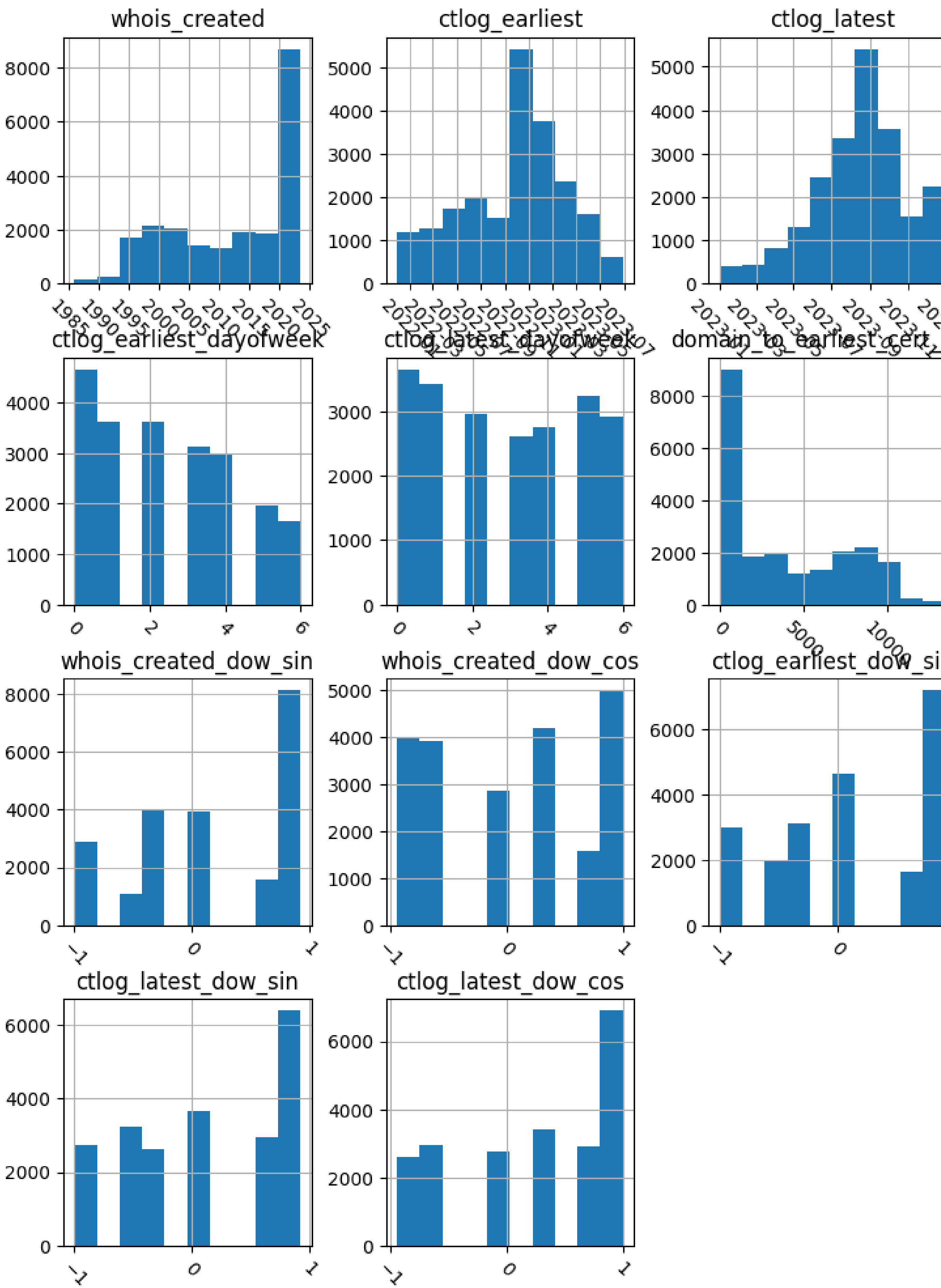

1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
whois_created_dow_cos
−1.0
−0.5
0.0
0.5
1.0
whois_created_dow_sin



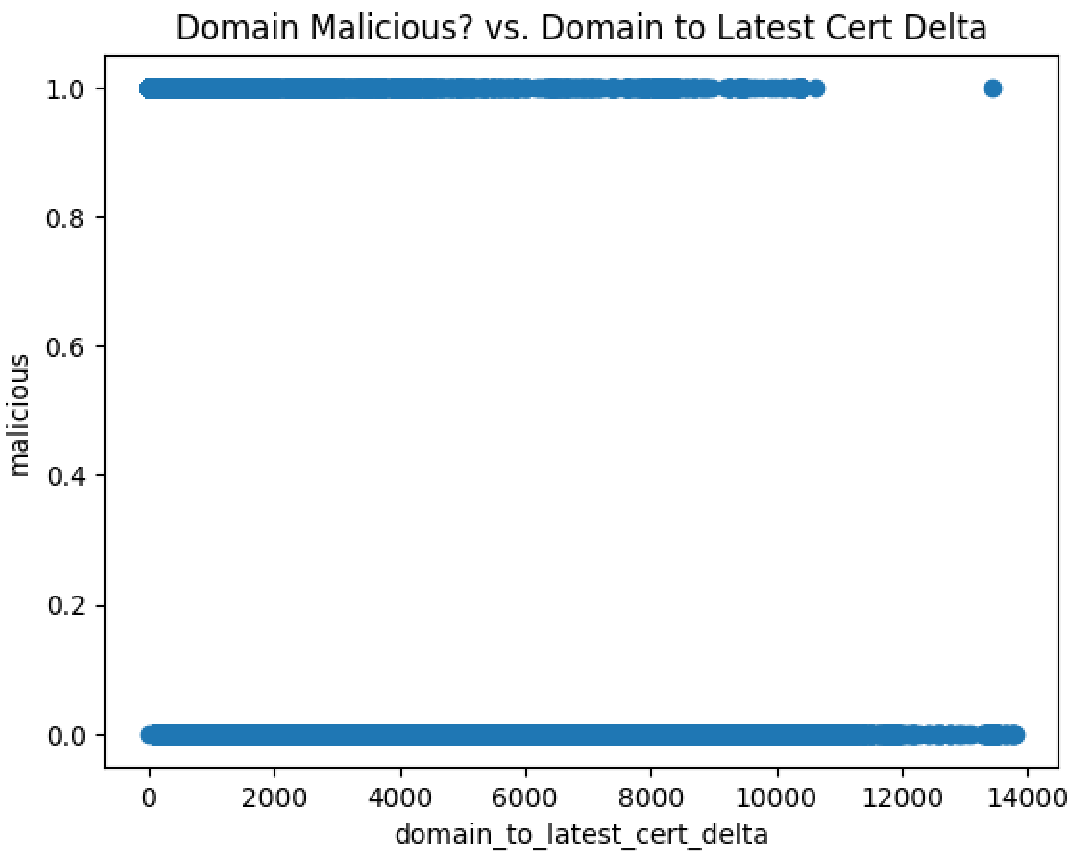
In [4]:

```
# Plot histograms
df.hist(figsize=(12,12), xrot=-45)

# Create a scatter plot of the data, showing malicious and benign domains
# plot the data as a scatter plot
plt.scatter(df["domain_to_earliest_cert_delta"], df["malicious"])
plt.xlabel("domain_to_earliest_cert_delta")
plt.ylabel("malicious")
plt.title("Domain Malicious? vs. Domain to Earliest Cert Delta")
plt.show()
# and for the latest
plt.scatter(df["domain_to_latest_cert_delta"], df["malicious"])
plt.xlabel("domain_to_latest_cert_delta")
plt.ylabel("malicious")
plt.title("Domain Malicious? vs. Domain to Latest Cert Delta")
plt.show()



click.echo(df.head())

# print a count of malicious and benign values
click.echo(df["malicious"].value_counts())
# deduplicate any rows, there shouldn't be any, but just in case
df = df.drop_duplicates()
click.echo(df["malicious"].value_counts())
```

```
click.echo(df.head())

X = df.drop(["malicious","domain", "ctlog_earliest", "ctlog_latest",
"whois_created", "whois_created_dayofweek", "ctlog_earliest_dayofweek"],
axis=1)
X = df.filter(combo_features)
y = df.filter(["malicious"])


click.echo(X.describe())
```



whois_created
ctlog_earliest
ctlog_latest
ctlog_earliest_dayofweek
ctlog_latest_dayofweek
whois_created_dow_sin
whois_created_dow_cos
ctlog_latest_dow_sin
ctlog_latest_dow_cos




```
                             domain  malicious       whois_created
0   i-db5p-cor001.api.p001.1drv.com      False 2013-08-05 18:33:50   \
4        soundcloud-pax.pandora.com      False 1993-12-28 05:00:00
5          joolcomercializadora.com       True 2023-05-22 14:53:50
6         createpdf-asr.acrobat.com      False 1999-03-16 05:00:00
8                           popt.in      False 2016-05-14 16:58:55

        ctlog_earliest         ctlog_latest  ctlog_wildcard
0 2022-01-24 20:01:58 2023-06-08 20:46:06            True   \
4 2022-05-19 00:00:00 2023-06-19 23:59:59            True
5 2022-04-06 22:23:24 2023-09-22 23:59:59           False
6 2022-09-09 00:00:00 2023-10-10 23:59:59            True
8 2023-01-07 20:36:15 2023-08-15 04:16:52           False

   whois_created_dayofweek  ctlog_earliest_dayofweek
ctlog_latest_dayofweek
0                        0                          0
3   \
4                        1                          3
0
5                        0                          2
4
6                        1                          4
1
8                        5                          5
1
```

```
   domain_to_earliest_cert_delta  domain_to_latest_cert_delta
0                         3095.0                       3595.0  \
4                        10369.0                      10766.0
5                          410.0                        124.0
6                         8578.0                       8975.0
8                         2430.0                       2649.0

   whois_created_dow_sin  whois_created_dow_cos  ctlog_earliest_dow_sin
0               0.000000               1.000000                0.000000  \
4               0.918032               0.396506               -0.340712
5               0.000000               1.000000                0.728010
6               0.918032               0.396506               -0.998199
8              -0.450871               0.892589               -0.450871

   ctlog_earliest_dow_cos  ctlog_latest_dow_sin  ctlog_latest_dow_cos
0                1.000000             -0.340712             -0.940168
4               -0.940168              0.000000              1.000000
5               -0.685567             -0.998199             -0.059997
6               -0.059997              0.918032              0.396506
8                0.892589              0.918032              0.396506
malicious
False    11739
True      9810
Name: count, dtype: int64
malicious
False    11739
True      9810
Name: count, dtype: int64
                            domain  malicious       whois_created
0  i-db5p-cor001.api.p001.1drv.com      False 2013-08-05 18:33:50  \
4       soundcloud-pax.pandora.com      False 1993-12-28 05:00:00
5         joolcomercializadora.com       True 2023-05-22 14:53:50
6        createpdf-asr.acrobat.com      False 1999-03-16 05:00:00
8                          popt.in      False 2016-05-14 16:58:55

       ctlog_earliest        ctlog_latest  ctlog_wildcard
0 2022-01-24 20:01:58 2023-06-08 20:46:06            True  \
4 2022-05-19 00:00:00 2023-06-19 23:59:59            True
5 2022-04-06 22:23:24 2023-09-22 23:59:59           False
6 2022-09-09 00:00:00 2023-10-10 23:59:59            True
8 2023-01-07 20:36:15 2023-08-15 04:16:52           False

   whois_created_dayofweek  ctlog_earliest_dayofweek
ctlog_latest_dayofweek
0                        0                         0
3  \
4                        1                         3
0
```

```
5                        0                          2
4
6                        1                          4
1
8                        5                          5
1

   domain_to_earliest_cert_delta  domain_to_latest_cert_delta
0                         3095.0                       3595.0  \
4                        10369.0                      10766.0
5                          410.0                        124.0
6                         8578.0                       8975.0
8                         2430.0                       2649.0

   whois_created_dow_sin  whois_created_dow_cos  ctlog_earliest_dow_sin
0               0.000000               1.000000                0.000000  \
4               0.918032               0.396506               -0.340712
5               0.000000               1.000000                0.728010
6               0.918032               0.396506               -0.998199
8              -0.450871               0.892589               -0.450871

   ctlog_earliest_dow_cos  ctlog_latest_dow_sin  ctlog_latest_dow_cos
0                1.000000             -0.340712             -0.940168
4               -0.940168              0.000000              1.000000
5               -0.685567             -0.998199             -0.059997
6               -0.059997              0.918032              0.396506
8                0.892589              0.918032              0.396506
       domain_to_earliest_cert_delta  ctlog_earliest_dow_sin
count                   21549.000000            21549.000000  \
mean                     3742.948397                0.095357
std                      3694.584062                0.651782
min                         0.000000               -0.998199
25%                       181.000000               -0.340712
50%                      2637.000000                0.000000
75%                      7078.000000                0.728010
max                     13445.000000                0.918032

       ctlog_earliest_dow_cos
count            21549.000000
mean                 0.161451
std                  0.734891
min                 -0.940168
25%                 -0.685567
50%                  0.396506
75%                  0.892589
max                  1.000000
```

In [5]:

```
# convert y (malicious) to 1/0 int
y = y.astype('int')
```

```
# convert X["ctlog_wildcard"]  to 1/0 int
if "ctlog_wildcard" in X.columns:
    X["ctlog_wildcard"] = X["ctlog_wildcard"].astype('int')

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

X_train = sm.add_constant(X_train)
X_test = sm.add_constant(X_test)

smfit = sm.Logit(y_train,X_train).fit()

smfit.summary()
Optimization terminated successfully.
         Current function value: 0.354501
         Iterations 7
```

Out[5]:

Logit Regression Results

Dep. Variable:	malicious	**No. Observations:**	17239
Model:	Logit	**Df Residuals:**	17234
Method:	MLE	**Df Model:**	4
Date:	Tue, 08 Aug 2023	**Pseudo R-squ.:**	0.4858
Time:	19:10:40	**Log-Likelihood:**	-6111.2
converged:	True	**LL-Null:**	-11885.
Covariance Type:	nonrobust	**LLR p-value:**	0.000

	coef	**std err**	**z**	**P>|z|**	**[0.025**	**0.975]**
const	2.1040	0.035	59.451	0.000	2.035	2.173
domain_to_earliest_cert_delta	-0.0006	9.92e-06	-61.542	0.000	-0.001	-0.001
ctlog_earliest_dow_sin	0.1371	0.035	3.902	0.000	0.068	0.206
ctlog_earliest_dow_cos	-0.1865	0.032	-5.911	0.000	-0.248	-0.125
ctlog_wildcard	-1.2052	0.049	-24.719	0.000	-1.301	-1.110

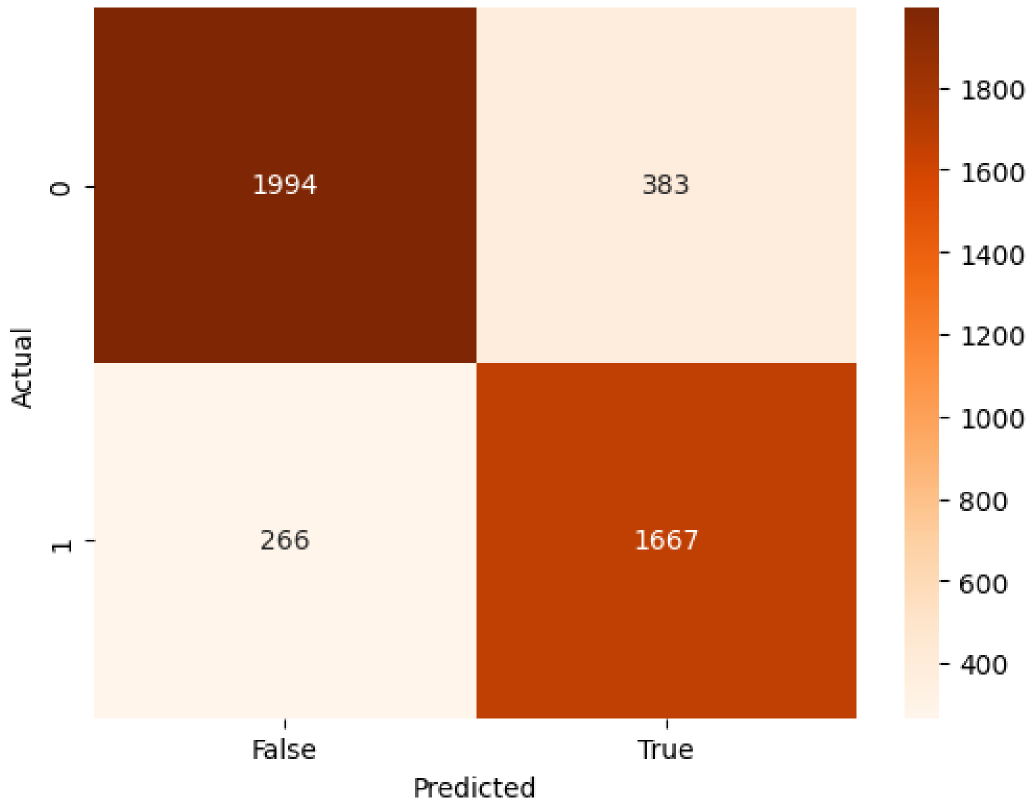
In [6]:

```
# Predict the malicious column using the test data
#add the incepts


y_predicted = smfit.predict(X_test)

# Present the results in a confusion matrix
confusion_matrix = confusion_matrix(y_test, y_predicted.round())
click.echo(confusion_matrix)

click.echo("Classification report:")
click.echo(classification_report(y_test, y_predicted.round()))

# Heatmap of confusion matrix
y_predicted
```

```
threshold = 0.5
y_predicted = [y > threshold for y in y_predicted]
data = {'Actual':    y_test.values.flatten(),
        'Predicted': y_predicted
        }

# Generate a confusion matrix and heatmap to evaluate the Type I and Type
II errors/ FP/FN etc.
df = pd.DataFrame(data, columns=['Actual','Predicted'])
confusion_matrix = pd.crosstab(df['Actual'], df['Predicted'],
rownames=['Actual'], colnames=['Predicted'])
fig = sns.heatmap(confusion_matrix, annot=True, cmap='Oranges', fmt='g')
fig
[[1994  383]
 [ 266 1667]]
Classification report:
              precision    recall  f1-score   support

           0       0.88      0.84      0.86      2377
           1       0.81      0.86      0.84      1933

    accuracy                           0.85      4310
   macro avg       0.85      0.85      0.85      4310
weighted avg       0.85      0.85      0.85      4310
```

Out[6]:

```
<Axes: xlabel='Predicted', ylabel='Actual'>
```



V. Feature Set D

In [1]:

```
import click
import pandas as pd
import glob
import os
#import sklearn
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report
from sklearn.preprocessing import StandardScaler
from scipy import stats
from sklearn.linear_model import LogisticRegression
import statsmodels.api as sm
import matplotlib.pyplot as plt
from datetime import datetime, date
# qqplot of the data
import seaborn as sns
import math
import numpy as np
import utils

combo_features =     ['domain_to_earliest_cert_delta',
'domain_to_latest_cert_delta']

path = "../data/"
filename = None

epoch = datetime(1970,1,1)
# open the training data file, from ../data/ for the most recent merged
training data file saved by prepare-training-data-parallel.py
full_path = path + "merged_20230705-104357_training_adorned-engineered.csv"
# get latest file matching the glob
if not filename:
    filename = max(glob.glob(full_path), key=os.path.getctime)
    click.echo(f"Using {filename} as training data")
    df = pd.read_csv(filename, sep=",")

# shuffle the rows
df = df.sample(frac=1, random_state=42).reset_index(drop=True)

click.echo(df.shape)
click.echo(df.columns)

""" # From prepare-training-data-parallel.py:
# Columns:
url
verification_time
domain
```

```
malicious
dateadded
whois_created
soa_timestamp
ctlog_earliest
ctlog_latest
ctlog_wildcard
whois_created_dayofweek,
ctlog_earliest_dayofweek,
domain_to_cert_delta
"""


# Data cleaning - there may be some epoch values in the whois_created
#  & ct_log_earliest columns, so we need to remove those rows

# convert the whois_created and ctlog_earliest, ctlog_latest columns to
datetime

df = df.loc[df["whois_created"] != ""]
df = df.loc[df["ctlog_earliest"] != ""]
df = df.loc[df["ctlog_latest"] != ""]

# some dates are have nanoseconds in them, so we need to remove the
nanosecond part
df["whois_created"] = df["whois_created"].str.split(".", n = 1, expand =
True)[0]
# some dates has additional timezone information, so we need to remove that
df["whois_created"] = df["whois_created"].apply(lambda x: " ".join(
str(x).split(" ")[:2]))


# convert the whois_created and ct_log_earliest columns to datetime
df = df.astype({"whois_created": 'datetime64[ns]', "ctlog_earliest":
'datetime64[ns]', "ctlog_latest": 'datetime64[ns]'})
df = df.loc[df["whois_created"].ne(pd.Timestamp('1970-01-01 00:00:00'))]
df = df.loc[df["ctlog_earliest"].ne(pd.Timestamp('1970-01-01 00:00:00'))]
df = df.loc[df["ctlog_latest"].ne(pd.Timestamp('1970-01-01 00:00:00'))]


# malicious is a bool
df['malicious'] = df['malicious'].astype('bool')
df['domain'] = df['domain'].astype('string')
```

```
Using ../data/merged_20230705-104357_training_adorned-engineered.csv as
training data
(35438, 13)
Index(['url', 'verification_time', 'domain', 'malicious', 'dateadded',
       'whois_created', 'soa_timestamp', 'ctlog_earliest', 'ctlog_latest',
```

```
       'ctlog_wildcard', 'whois_created_dayofweek',
'ctlog_earliest_dayofweek',
       'domain_to_cert_delta'],
      dtype='object')
```

In [2]:

```
#########
# Feature engineering
#########

# remove any rows where the whois_created or ctlog_earliest columns are
null
df = df.loc[df["whois_created"].notnull()]
df = df.loc[df["ctlog_earliest"].notnull()]

# if the data is missing the "whois_created_dayofweek" or
"ctlog_earliest_dayofweek" columns, then add them
# whois created day of the week 0 = Monday, 6 = Sunday
df["whois_created_dayofweek"] = df["whois_created"].apply(
        lambda x: x.weekday() if isinstance(x, date) else None
    )

# cert valid day of the week 0 = Monday, 6 = Sunday
df["ctlog_earliest_dayofweek"] = df["ctlog_earliest"].apply(
        lambda x: x.weekday() if isinstance(x, date) else None
    )

df["ctlog_latest_dayofweek"] = df["ctlog_latest"].apply(
        lambda x: x.weekday() if isinstance(x, date) else None
    )

# set data type of the day of week columns to int
df["whois_created_dayofweek"] =
df["whois_created_dayofweek"].astype("int64")
df["ctlog_earliest_dayofweek"] =
df["ctlog_earliest_dayofweek"].astype("int64")
df["ctlog_latest_dayofweek"] = df["ctlog_latest_dayofweek"].astype("int64")

# domain to cert delta
df["domain_to_earliest_cert_delta"] = df.apply(
        lambda x: utils.get_days_delta(
            x["ctlog_earliest"],
            x["whois_created"]
            if x["whois_created"] and x["ctlog_earliest"]
            else None,
        ),
        axis=1,
    )

# set the data type of the domain_to_cert_delta column to float
```

```
df["domain_to_earliest_cert_delta"] =
df["domain_to_earliest_cert_delta"].astype("float64")

# domain to latest cert delta
df["domain_to_latest_cert_delta"] = df.apply(
        lambda x: utils.get_days_delta(
            x["ctlog_latest"],
            x["whois_created"]
            if x["whois_created"] and x["ctlog_latest"]
            else None,
        ),
        axis=1,
    )

# set the data type of the domain_to_cert_delta column to float
df["domain_to_latest_cert_delta"] =
df["domain_to_latest_cert_delta"].astype("float64")

# drop columns we imported that we will never use
df = df.drop(columns=["url", "verification_time", "dateadded",
"soa_timestamp","domain_to_cert_delta"])

click.echo(df.head())
click.echo(df.describe(include='all'))
click.echo(df.dtypes)
                            domain  malicious       whois_created
0  i-db5p-cor001.api.p001.1drv.com      False 2013-08-05 18:33:50  \
4       soundcloud-pax.pandora.com      False 1993-12-28 05:00:00
5         joolcomercializadora.com       True 2023-05-22 14:53:50
6        createpdf-asr.acrobat.com      False 1999-03-16 05:00:00
8                          popt.in      False 2016-05-14 16:58:55

       ctlog_earliest        ctlog_latest  ctlog_wildcard
0 2022-01-24 20:01:58 2023-06-08 20:46:06            True  \
4 2022-05-19 00:00:00 2023-06-19 23:59:59            True
5 2022-04-06 22:23:24 2023-09-22 23:59:59           False
6 2022-09-09 00:00:00 2023-10-10 23:59:59            True
8 2023-01-07 20:36:15 2023-08-15 04:16:52           False

   whois_created_dayofweek  ctlog_earliest_dayofweek
ctlog_latest_dayofweek
0                        0                          0
3  \
4                        1                          3
0
5                        0                          2
4
6                        1                          4
1
```

```
8                        5                          5
1

   domain_to_earliest_cert_delta  domain_to_latest_cert_delta
0                        -3095.0                      -3595.0
4                       -10369.0                     -10766.0
5                          410.0                       -124.0
6                        -8578.0                      -8975.0
8                        -2430.0                      -2649.0
                  domain malicious                  whois_created
count              21549     21549                          21549  \
unique             21536         2                            NaN
top     www.mediafire.com    False                            NaN
freq                   2     11739                            NaN
mean                 NaN       NaN  2012-10-03 12:56:32.335050496
min                  NaN       NaN            1986-01-09 00:00:00
25%                  NaN       NaN            2003-05-25 13:35:05
50%                  NaN       NaN            2015-05-07 23:56:05
75%                  NaN       NaN            2023-03-20 15:03:16
max                  NaN       NaN            2023-07-03 08:21:24
std                  NaN       NaN                            NaN

                        ctlog_earliest                   ctlog_latest
count                            21549                          21549  \
unique                             NaN                            NaN
top                                NaN                            NaN
freq                               NaN                            NaN
mean    2022-09-26 15:45:50.943570432  2023-08-14 17:49:06.400900352
min               2021-11-30 05:24:28            2023-01-01 18:42:11
25%               2022-06-24 13:47:12            2023-07-02 08:11:07
50%               2022-10-18 21:00:14            2023-08-21 21:40:11
75%               2022-12-14 00:00:00            2023-09-21 19:41:38
max               2023-06-28 04:36:22            2023-12-31 23:59:59
std                                NaN                            NaN

       ctlog_wildcard  whois_created_dayofweek  ctlog_earliest_dayofweek
count           21549             21549.000000              21549.000000  \
unique              2                      NaN                       NaN
top             False                      NaN                       NaN
freq            13032                      NaN                       NaN
mean              NaN                 2.332823                  2.399462
min               NaN                 0.000000                  0.000000
25%               NaN                 1.000000                  1.000000
50%               NaN                 2.000000                  2.000000
75%               NaN                 4.000000                  4.000000
max               NaN                 6.000000                  6.000000
std               NaN                 1.775043                  1.897252

        ctlog_latest_dayofweek  domain_to_earliest_cert_delta
```

```
count             21549.000000                  21549.000000  \
unique                     NaN                           NaN
top                        NaN                           NaN
freq                       NaN                           NaN
mean                  2.873080                  -3645.602070
min                   0.000000                 -13445.000000
25%                   1.000000                  -7078.000000
50%                   3.000000                  -2637.000000
75%                   5.000000                     69.000000
max                   6.000000                    524.000000
std                   2.057394                   3790.677119

        domain_to_latest_cert_delta
count                  21549.000000
unique                          NaN
top                             NaN
freq                            NaN
mean                   -3967.678222
min                   -13798.000000
25%                    -7421.000000
50%                    -3009.000000
75%                     -144.000000
max                      135.000000
std                     3852.703681
domain                               string[python]
malicious                                      bool
whois_created                        datetime64[ns]
ctlog_earliest                       datetime64[ns]
ctlog_latest                         datetime64[ns]
ctlog_wildcard                                 bool
whois_created_dayofweek                       int64
ctlog_earliest_dayofweek                      int64
ctlog_latest_dayofweek                        int64
domain_to_earliest_cert_delta               float64
domain_to_latest_cert_delta                 float64
dtype: object
```

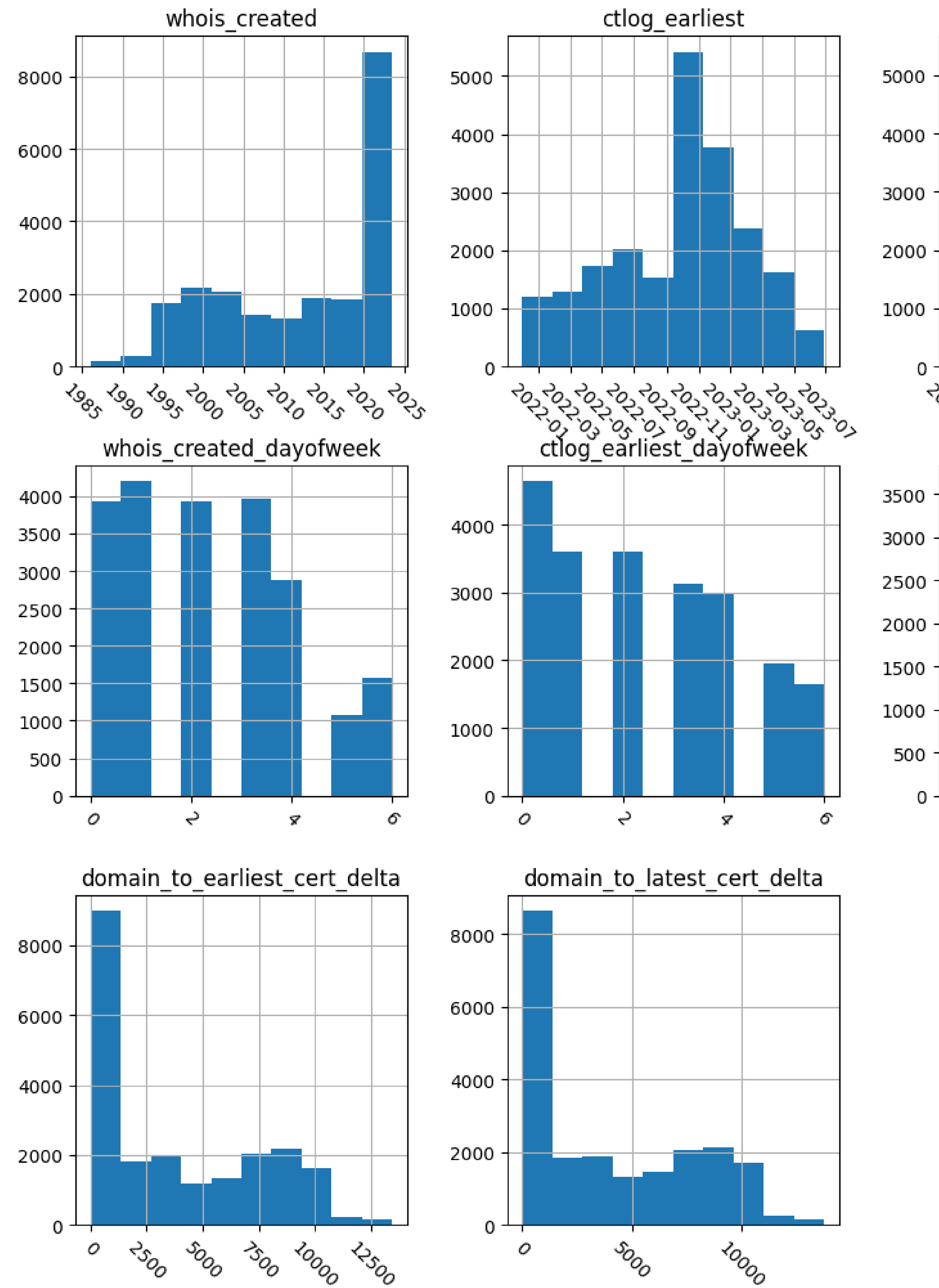
In [3]:

```
# absolute value of the domain_to_cert_delta
df["domain_to_earliest_cert_delta"] =
abs(df["domain_to_earliest_cert_delta"])
df["domain_to_latest_cert_delta"] = abs(df["domain_to_latest_cert_delta"])

df.hist(figsize=(12,12), xrot=-45)

col_index = df.columns.get_loc("whois_created_dayofweek")
df["whois_created_dow_sin"] = df.apply(lambda row:
math.sin(360/7*(row[col_index])),axis=1)
df["whois_created_dow_cos"] = df.apply(lambda row:
math.cos(360/7*(row[col_index])),axis=1)
```

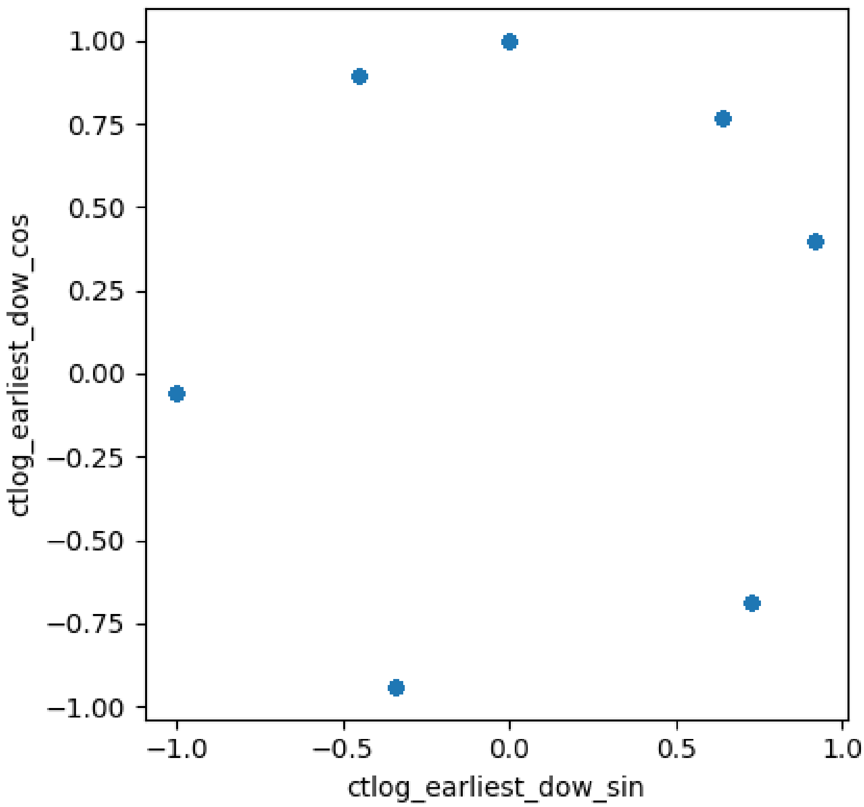
```
col_index = df.columns.get_loc("ctlog_earliest_dayofweek")
df["ctlog_earliest_dow_sin"] = df.apply(lambda row:
math.sin(360/7*(row[col_index])),axis=1)
df["ctlog_earliest_dow_cos"] = df.apply(lambda row:
math.cos(360/7*(row[col_index])),axis=1)

col_index = df.columns.get_loc("ctlog_latest_dayofweek")
df["ctlog_latest_dow_sin"] = df.apply(lambda row:
math.sin(360/7*(row[col_index])),axis=1)
df["ctlog_latest_dow_cos"] = df.apply(lambda row:
math.cos(360/7*(row[col_index])),axis=1)


df.plot.scatter(x="ctlog_earliest_dow_sin",
y="ctlog_earliest_dow_cos").set_aspect('equal')
df.plot.scatter(x="whois_created_dow_sin",
y="whois_created_dow_cos").set_aspect('equal')
df.plot.scatter(x="ctlog_latest_dow_sin",
y="ctlog_latest_dow_cos").set_aspect('equal')

"""
# add one hot encode ctlog_earliest_not_before_dayofweek
df = pd.get_dummies(
        df,
        columns=["ctlog_earliest_dayofweek"],
        prefix=["ctlog_earliest_dow"],
    )
# add one hot encode whois_created_dayofweek to columns
df = pd.get_dummies(
        df, columns=["whois_created_dayofweek"], prefix="whois_dow"
    )
    """

# Summary statistics
click.echo(df.describe(include='all'))
                   domain malicious                   whois_created
count               21549     21549                           21549  \
unique              21536         2                             NaN
top     www.mediafire.com     False                             NaN
freq                    2     11739                             NaN
mean                  NaN       NaN  2012-10-03 12:56:32.335050496
min                   NaN       NaN            1986-01-09 00:00:00
25%                   NaN       NaN            2003-05-25 13:35:05
50%                   NaN       NaN            2015-05-07 23:56:05
75%                   NaN       NaN            2023-03-20 15:03:16
max                   NaN       NaN            2023-07-03 08:21:24
std                   NaN       NaN                             NaN
```

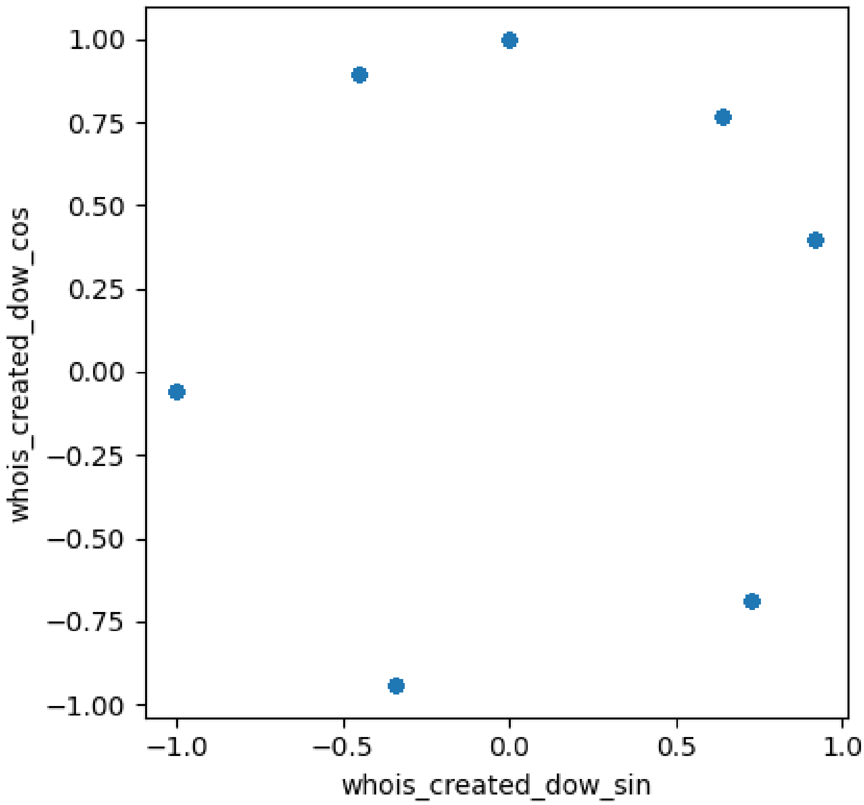
```
                     ctlog_earliest                   ctlog_latest
count                         21549                          21549  \
unique                          NaN                            NaN
top                             NaN                            NaN
freq                            NaN                            NaN
mean    2022-09-26 15:45:50.943570432  2023-08-14 17:49:06.400900352
min               2021-11-30 05:24:28            2023-01-01 18:42:11
25%               2022-06-24 13:47:12            2023-07-02 08:11:07
50%               2022-10-18 21:00:14            2023-08-21 21:40:11
75%               2022-12-14 00:00:00            2023-09-21 19:41:38
max               2023-06-28 04:36:22            2023-12-31 23:59:59
std                             NaN                            NaN

       ctlog_wildcard  whois_created_dayofweek  ctlog_earliest_dayofweek
count           21549             21549.000000              21549.000000  \
unique              2                      NaN                       NaN
top             False                      NaN                       NaN
freq            13032                      NaN                       NaN
mean              NaN                 2.332823                  2.399462
min               NaN                 0.000000                  0.000000
25%               NaN                 1.000000                  1.000000
50%               NaN                 2.000000                  2.000000
75%               NaN                 4.000000                  4.000000
max               NaN                 6.000000                  6.000000
std               NaN                 1.775043                  1.897252

        ctlog_latest_dayofweek  domain_to_earliest_cert_delta
count             21549.000000                   21549.000000  \
unique                     NaN                            NaN
top                        NaN                            NaN
freq                       NaN                            NaN
mean                  2.873080                    3742.948397
min                   0.000000                       0.000000
25%                   1.000000                     181.000000
50%                   3.000000                    2637.000000
75%                   5.000000                    7078.000000
max                   6.000000                   13445.000000
std                   2.057394                    3694.584062

        domain_to_latest_cert_delta  whois_created_dow_sin
count                  21549.000000           21549.000000  \
unique                          NaN                    NaN
top                             NaN                    NaN
freq                            NaN                    NaN
mean                    3969.491206               0.140419
min                        0.000000              -0.998199
25%                      144.000000              -0.340712
50%                     3009.000000               0.000000
75%                     7421.000000               0.728010
```

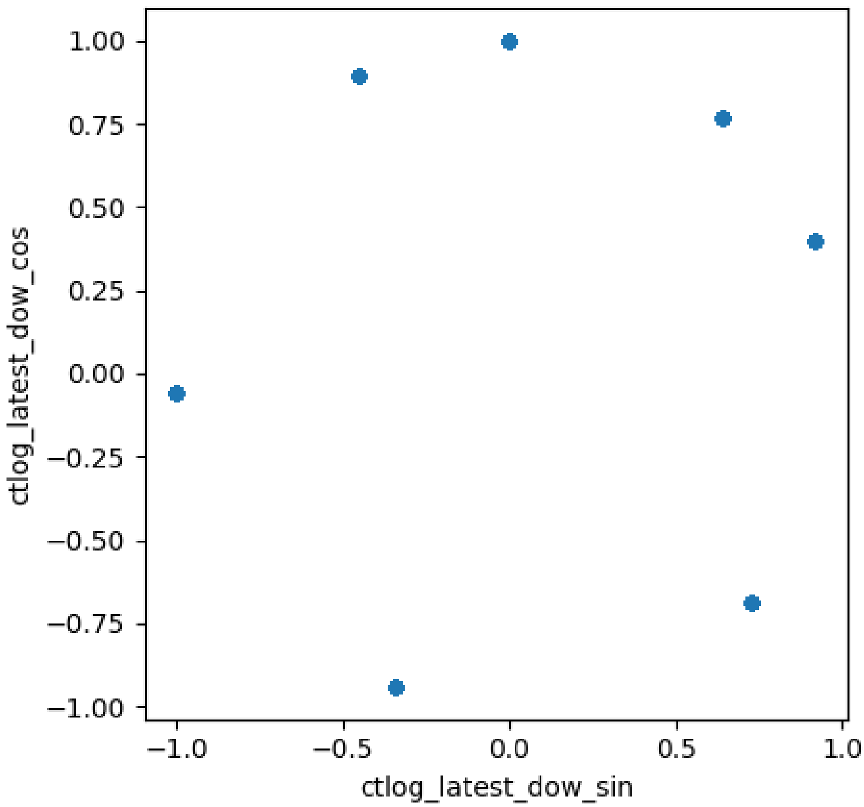
```
max                   13798.000000               0.918032
std                    3850.835626               0.659922

        whois_created_dow_cos  ctlog_earliest_dow_sin
ctlog_earliest_dow_cos
count             21549.000000            21549.000000
21549.000000  \
unique                     NaN                     NaN
NaN
top                        NaN                     NaN
NaN
freq                       NaN                     NaN
NaN
mean                  0.054288                0.095357
0.161451
min                  -0.940168               -0.998199               -
0.940168
25%                  -0.685567               -0.340712               -
0.685567
50%                   0.396506                0.000000
0.396506
75%                   0.767830                0.728010
0.892589
max                   1.000000                0.918032
1.000000
std                   0.736128                0.651782
0.734891

        ctlog_latest_dow_sin  ctlog_latest_dow_cos
count           21549.000000          21549.000000
unique                   NaN                   NaN
top                      NaN                   NaN
freq                     NaN                   NaN
mean                0.096253              0.255578
min                -0.998199             -0.940168
25%                -0.450871             -0.685567
50%                 0.000000              0.396506
75%                 0.728010              0.892589
max                 0.918032              1.000000
std                 0.651597              0.707728
```



whois_created
0
2000
4000
6000
8000
1985
1990
1995
2000
2005
2010
2015
2020
2025
ctlog_earliest
0
1000
2000
3000
4000
5000
2022-01
2022-03
2022-05
2022-07
2022-09
2022-11
2023-01
2023-03
2023-05
2023-07
whois_created_dayofweek
0
500
1000
1500
2000
2500
3000
3500
4000
0
2
4
6
ctlog_earliest_dayofweek
0
1000
2000
3000
4000
0
2
4
6
domain_to_earliest_cert_delta
0
2000
4000
6000
8000
0
2500
5000
7500
10000
12500
domain_to_latest_cert_delta
0
2000
4000
6000
8000
0
5000
10000


1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
ctlog_earliest_dow_cos
−1.0
−0.5
0.0
0.5
1.0
ctlog_earliest_dow_sin

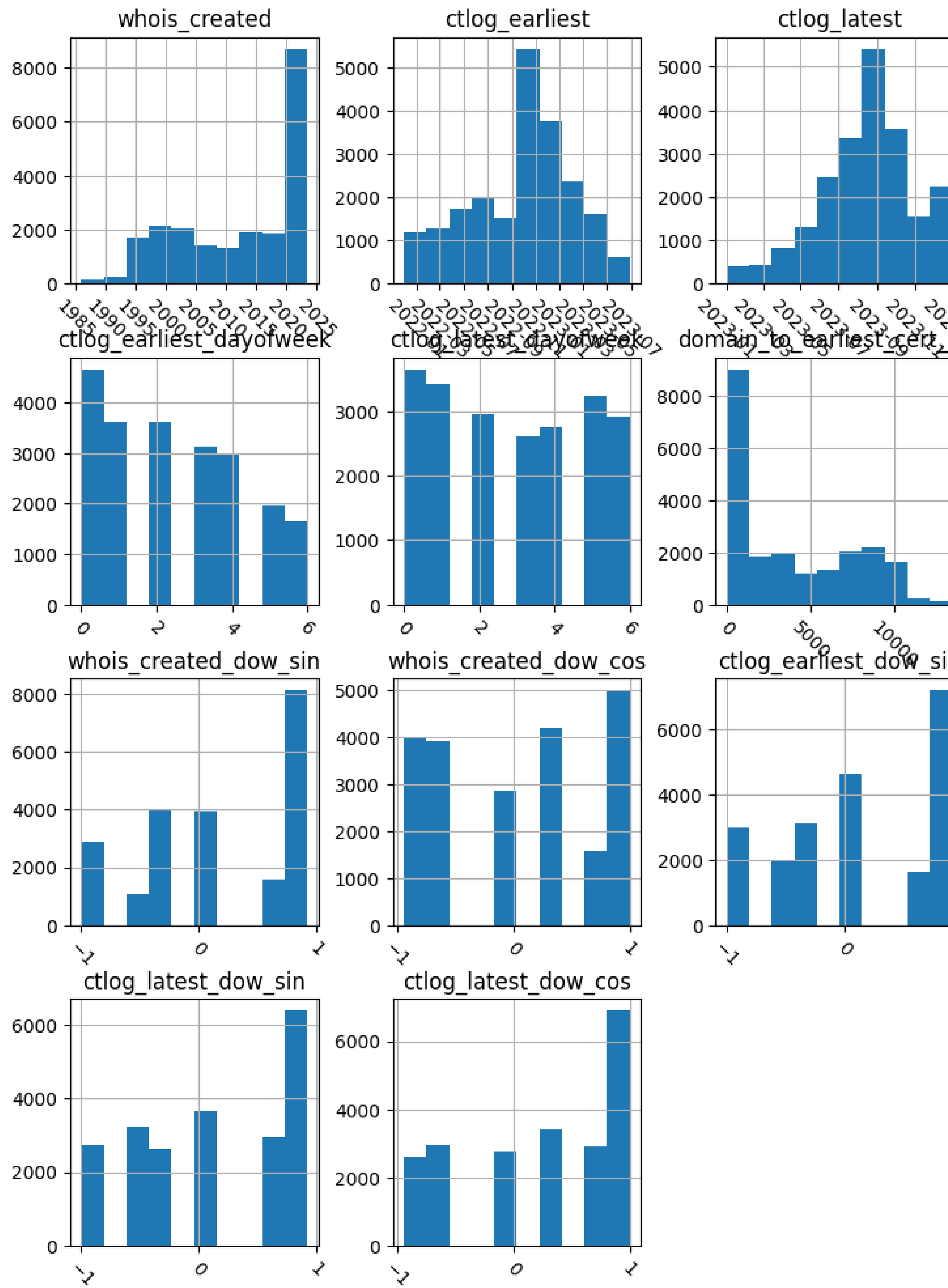

1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
whois_created_dow_cos
−1.0
−0.5
0.0
0.5
1.0
whois_created_dow_sin

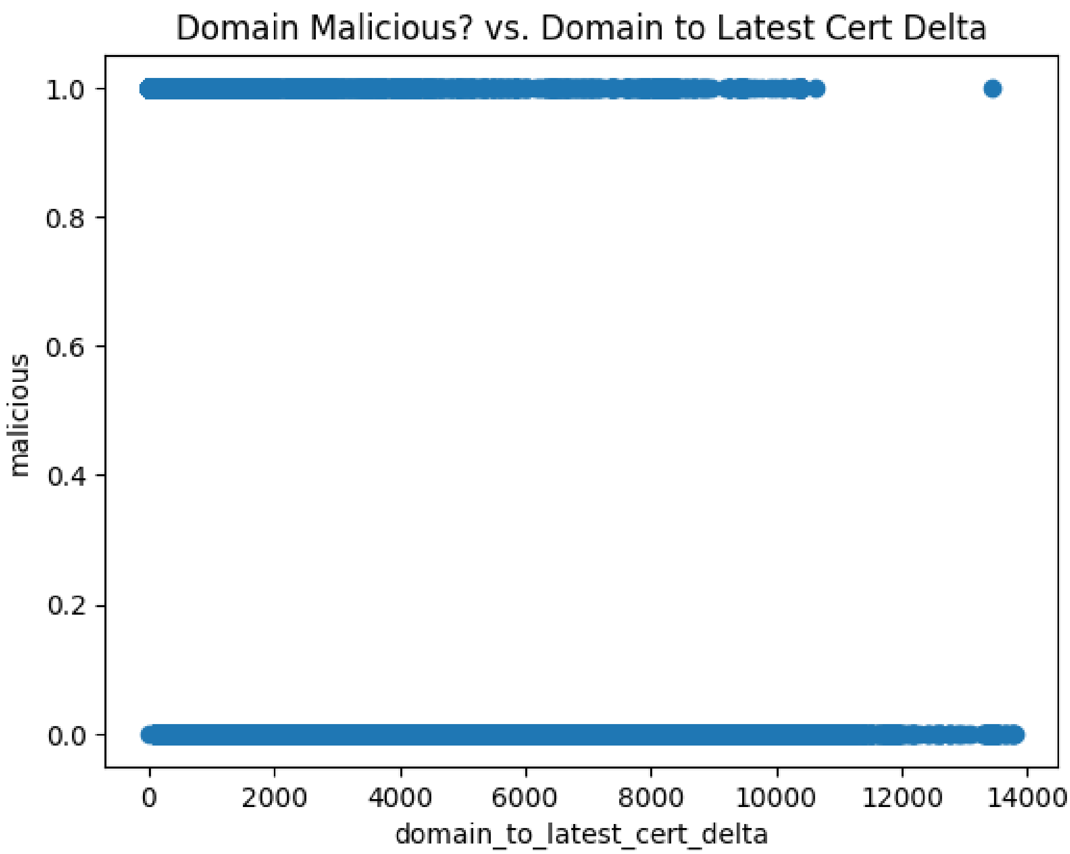


In [4]:

```
# Plot histograms
df.hist(figsize=(12,12), xrot=-45)

# Create a scatter plot of the data, showing malicious and benign domains
# plot the data as a scatter plot
plt.scatter(df["domain_to_earliest_cert_delta"], df["malicious"])
plt.xlabel("domain_to_earliest_cert_delta")
plt.ylabel("malicious")
plt.title("Domain Malicious? vs. Domain to Earliest Cert Delta")
plt.show()
# and for the latest
plt.scatter(df["domain_to_latest_cert_delta"], df["malicious"])
plt.xlabel("domain_to_latest_cert_delta")
plt.ylabel("malicious")
plt.title("Domain Malicious? vs. Domain to Latest Cert Delta")
plt.show()



click.echo(df.head())

# print a count of malicious and benign values
click.echo(df["malicious"].value_counts())
# deduplicate any rows, there shouldn't be any, but just in case
df = df.drop_duplicates()
click.echo(df["malicious"].value_counts())
```

```
click.echo(df.head())

X = df.drop(["malicious","domain", "ctlog_earliest", "ctlog_latest",
"whois_created", "whois_created_dayofweek", "ctlog_earliest_dayofweek"],
axis=1)
X = df.filter(combo_features)
y = df.filter(["malicious"])


click.echo(X.describe())
```


whois_created
ctlog_earliest
ctlog_latest
ctlog_earliest_dayofweek
ctlog_latest_dayofweek
domain_to_earliest_cert_
whois_created_dow_sin
whois_created_dow_cos
ctlog_earliest_dow_si
ctlog_latest_dow_sin
ctlog_latest_dow_cos
1985 1990 1995 2000 2005 2010 2015 2020 2025
0 2000 4000 6000 8000
0 1000 2000 3000 4000 5000
0 2 4 6
0 5000 10000
-1 0 1




```
                              domain  malicious       whois_created
0  i-db5p-cor001.api.p001.1drv.com      False 2013-08-05 18:33:50   \
4       soundcloud-pax.pandora.com      False 1993-12-28 05:00:00
5         joolcomercializadora.com       True 2023-05-22 14:53:50
6        createpdf-asr.acrobat.com      False 1999-03-16 05:00:00
8                          popt.in      False 2016-05-14 16:58:55

       ctlog_earliest        ctlog_latest  ctlog_wildcard
0 2022-01-24 20:01:58 2023-06-08 20:46:06            True   \
4 2022-05-19 00:00:00 2023-06-19 23:59:59            True
5 2022-04-06 22:23:24 2023-09-22 23:59:59           False
6 2022-09-09 00:00:00 2023-10-10 23:59:59            True
8 2023-01-07 20:36:15 2023-08-15 04:16:52           False

   whois_created_dayofweek  ctlog_earliest_dayofweek
ctlog_latest_dayofweek
0                        0                          0
3  \
4                        1                          3
0
5                        0                          2
4
6                        1                          4
1
8                        5                          5
1
```

```
   domain_to_earliest_cert_delta  domain_to_latest_cert_delta
0                         3095.0                       3595.0  \
4                        10369.0                      10766.0
5                          410.0                        124.0
6                         8578.0                       8975.0
8                         2430.0                       2649.0

   whois_created_dow_sin  whois_created_dow_cos  ctlog_earliest_dow_sin
0               0.000000               1.000000                0.000000  \
4               0.918032               0.396506               -0.340712
5               0.000000               1.000000                0.728010
6               0.918032               0.396506               -0.998199
8              -0.450871               0.892589               -0.450871

   ctlog_earliest_dow_cos  ctlog_latest_dow_sin  ctlog_latest_dow_cos
0                1.000000             -0.340712             -0.940168
4               -0.940168              0.000000              1.000000
5               -0.685567             -0.998199             -0.059997
6               -0.059997              0.918032              0.396506
8                0.892589              0.918032              0.396506
malicious
False    11739
True      9810
Name: count, dtype: int64
malicious
False    11739
True      9810
Name: count, dtype: int64
                             domain  malicious       whois_created
0  i-db5p-cor001.api.p001.1drv.com      False 2013-08-05 18:33:50  \
4       soundcloud-pax.pandora.com      False 1993-12-28 05:00:00
5         joolcomercializadora.com       True 2023-05-22 14:53:50
6        createpdf-asr.acrobat.com      False 1999-03-16 05:00:00
8                          popt.in      False 2016-05-14 16:58:55

       ctlog_earliest        ctlog_latest  ctlog_wildcard
0 2022-01-24 20:01:58 2023-06-08 20:46:06            True  \
4 2022-05-19 00:00:00 2023-06-19 23:59:59            True
5 2022-04-06 22:23:24 2023-09-22 23:59:59           False
6 2022-09-09 00:00:00 2023-10-10 23:59:59            True
8 2023-01-07 20:36:15 2023-08-15 04:16:52           False

   whois_created_dayofweek  ctlog_earliest_dayofweek
ctlog_latest_dayofweek
0                        0                         0
3  \
4                        1                         3
0
```

```
5                        0                          2
4
6                        1                          4
1
8                        5                          5
1

   domain_to_earliest_cert_delta  domain_to_latest_cert_delta
0                         3095.0                       3595.0  \
4                        10369.0                      10766.0
5                          410.0                        124.0
6                         8578.0                       8975.0
8                         2430.0                       2649.0

   whois_created_dow_sin  whois_created_dow_cos  ctlog_earliest_dow_sin
0               0.000000               1.000000                0.000000  \
4               0.918032               0.396506               -0.340712
5               0.000000               1.000000                0.728010
6               0.918032               0.396506               -0.998199
8              -0.450871               0.892589               -0.450871

   ctlog_earliest_dow_cos  ctlog_latest_dow_sin  ctlog_latest_dow_cos
0                1.000000             -0.340712             -0.940168
4               -0.940168              0.000000              1.000000
5               -0.685567             -0.998199             -0.059997
6               -0.059997              0.918032              0.396506
8                0.892589              0.918032              0.396506
       domain_to_earliest_cert_delta  domain_to_latest_cert_delta
count                   21549.000000                 21549.000000
mean                     3742.948397                  3969.491206
std                      3694.584062                  3850.835626
min                         0.000000                     0.000000
25%                       181.000000                   144.000000
50%                      2637.000000                  3009.000000
75%                      7078.000000                  7421.000000
max                     13445.000000                 13798.000000
```

In [5]:

```
# convert y (malicious) to 1/0 int
y = y.astype('int')
# convert X["ctlog_wildcard"]  to 1/0 int
if "ctlog_wildcard" in X.columns:
    X["ctlog_wildcard"] = X["ctlog_wildcard"].astype('int')

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

X_train = sm.add_constant(X_train)
X_test = sm.add_constant(X_test)
```

```
smfit = sm.Logit(y_train,X_train).fit()

smfit.summary()
Optimization terminated successfully.
         Current function value: 0.289776
         Iterations 7
```

Out[5]:

Logit Regression Results

Dep. Variable:	malicious	**No. Observations:**	17239
Model:	Logit	**Df Residuals:**	17236
Method:	MLE	**Df Model:**	2
Date:	Tue, 08 Aug 2023	**Pseudo R-squ.:**	0.5797
Time:	19:09:48	**Log-Likelihood:**	-4995.5
converged:	True	**LL-Null:**	-11885.
Covariance Type:	nonrobust	**LLR p-value:**	0.000

	coef	**std err**	**z**	**P>|z|**	**[0.025**	**0.975]**
const	3.0029	0.053	56.462	0.000	2.899	3.107
domain_to_earliest_cert_delta	0.0077	0.000	42.137	0.000	0.007	0.008
domain_to_latest_cert_delta	-0.0081	0.000	-45.030	0.000	-0.008	-0.008

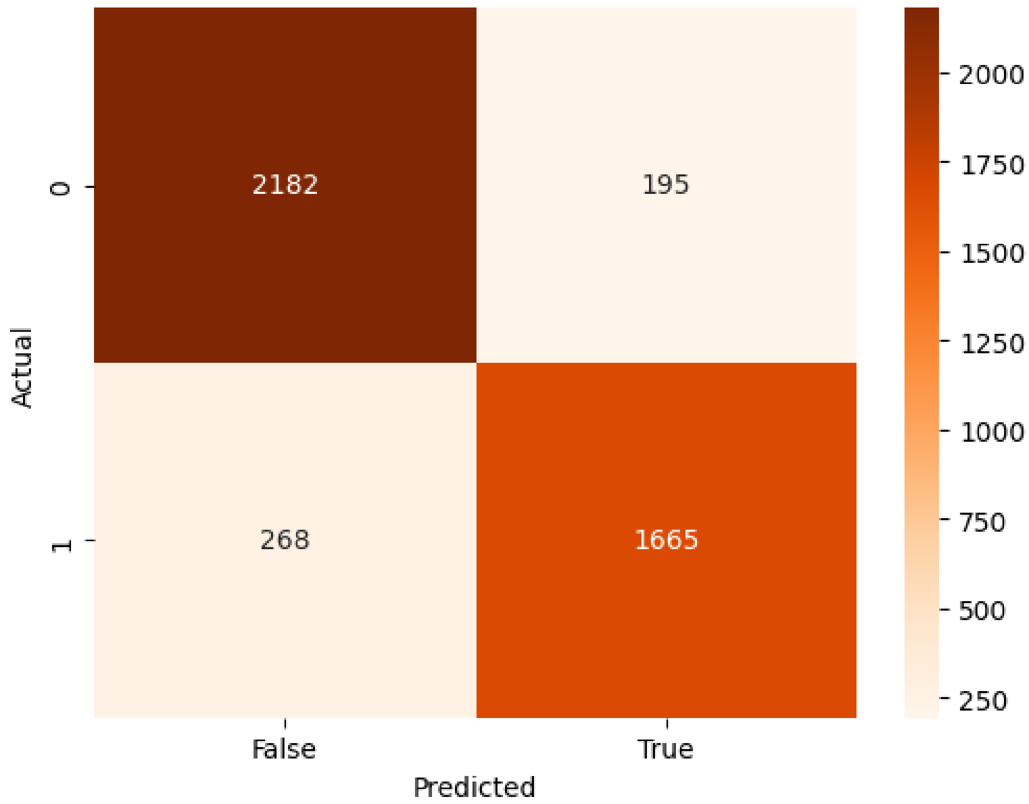
In [6]:

```
# Predict the malicious column using the test data
#add the incepts


y_predicted = smfit.predict(X_test)

# Present the results in a confusion matrix
confusion_matrix = confusion_matrix(y_test, y_predicted.round())
click.echo(confusion_matrix)

click.echo("Classification report:")
click.echo(classification_report(y_test, y_predicted.round()))

# Heatmap of confusion matrix
y_predicted

threshold = 0.5
y_predicted = [y > threshold for y in y_predicted]
data = {'Actual':    y_test.values.flatten(),
        'Predicted': y_predicted
        }

# Generate a confusion matrix and heatmap to evaluate the Type I and Type
II errors/ FP/FN etc.
df = pd.DataFrame(data, columns=['Actual','Predicted'])
confusion_matrix = pd.crosstab(df['Actual'], df['Predicted'],
rownames=['Actual'], colnames=['Predicted'])
fig = sns.heatmap(confusion_matrix, annot=True, cmap='Oranges', fmt='g')
```

```
fig
[[2182  195]
 [ 268 1665]]
Classification report:
              precision    recall  f1-score   support

           0       0.89      0.92      0.90      2377
           1       0.90      0.86      0.88      1933

    accuracy                           0.89      4310
   macro avg       0.89      0.89      0.89      4310
weighted avg       0.89      0.89      0.89      4310
```

Out[6]:

```
<Axes: xlabel='Predicted', ylabel='Actual'>
```



VI. Feature Set E

In [1]:

```
import click
import pandas as pd
import glob
import os
#import sklearn
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report
from sklearn.preprocessing import StandardScaler
from scipy import stats
from sklearn.linear_model import LogisticRegression
import statsmodels.api as sm
import matplotlib.pyplot as plt
from datetime import datetime, date
# qqplot of the data
import seaborn as sns
import math
import numpy as np
import utils

combo_features = [
    'domain_to_earliest_cert_delta',
    'domain_to_latest_cert_delta',
    'ctlog_earliest_dow_sin',
    'ctlog_earliest_dow_cos',
    'ctlog_latest_dow_sin',
    'ctlog_latest_dow_cos',
    'ctlog_wildcard'
    ]

path = "../data/"
filename = None

epoch = datetime(1970,1,1)
# open the training data file, from ../data/ for the most recent merged
training data file saved by prepare-training-data-parallel.py
full_path = path + "merged_20230705-104357_training_adorned-engineered.csv"
# get latest file matching the glob
if not filename:
    filename = max(glob.glob(full_path), key=os.path.getctime)
    click.echo(f"Using {filename} as training data")
    df = pd.read_csv(filename, sep=",")

# shuffle the rows
df = df.sample(frac=1, random_state=42).reset_index(drop=True)

click.echo(df.shape)
```

```
click.echo(df.columns)

""" # From prepare-training-data-parallel.py:
# Columns:
url
verification_time
domain
malicious
dateadded
whois_created
soa_timestamp
ctlog_earliest
ctlog_latest
ctlog_wildcard
whois_created_dayofweek,
ctlog_earliest_dayofweek,
domain_to_cert_delta
"""


# Data cleaning - there may be some epoch values in the whois_created
#  & ct_log_earliest columns, so we need to remove those rows

# convert the whois_created and ctlog_earliest, ctlog_latest columns to
datetime

df = df.loc[df["whois_created"] != ""]
df = df.loc[df["ctlog_earliest"] != ""]
df = df.loc[df["ctlog_latest"] != ""]

# some dates are have nanoseconds in them, so we need to remove the
nanosecond part
df["whois_created"] = df["whois_created"].str.split(".", n = 1, expand =
True)[0]
# some dates has additional timezone information, so we need to remove that
df["whois_created"] = df["whois_created"].apply(lambda x: " ".join(
str(x).split(" ")[:2]))


# convert the whois_created and ct_log_earliest columns to datetime
df = df.astype({"whois_created": 'datetime64[ns]', "ctlog_earliest":
'datetime64[ns]', "ctlog_latest": 'datetime64[ns]'})
df = df.loc[df["whois_created"].ne(pd.Timestamp('1970-01-01 00:00:00'))]
df = df.loc[df["ctlog_earliest"].ne(pd.Timestamp('1970-01-01 00:00:00'))]
df = df.loc[df["ctlog_latest"].ne(pd.Timestamp('1970-01-01 00:00:00'))]


# malicious is a bool
df['malicious'] = df['malicious'].astype('bool')
```

```
df['domain'] = df['domain'].astype('string')
Using ../data/merged_20230705-104357_training_adorned-engineered.csv as
training data
(35438, 13)
Index(['url', 'verification_time', 'domain', 'malicious', 'dateadded',
       'whois_created', 'soa_timestamp', 'ctlog_earliest', 'ctlog_latest',
       'ctlog_wildcard', 'whois_created_dayofweek',
'ctlog_earliest_dayofweek',
       'domain_to_cert_delta'],
      dtype='object')
```

In [2]:

```
#########
# Feature engineering
#########

# remove any rows where the whois_created or ctlog_earliest columns are
null
df = df.loc[df["whois_created"].notnull()]
df = df.loc[df["ctlog_earliest"].notnull()]

# if the data is missing the "whois_created_dayofweek" or
"ctlog_earliest_dayofweek" columns, then add them
# whois created day of the week 0 = Monday, 6 = Sunday
df["whois_created_dayofweek"] = df["whois_created"].apply(
        lambda x: x.weekday() if isinstance(x, date) else None
    )

# cert valid day of the week 0 = Monday, 6 = Sunday
df["ctlog_earliest_dayofweek"] = df["ctlog_earliest"].apply(
        lambda x: x.weekday() if isinstance(x, date) else None
    )

df["ctlog_latest_dayofweek"] = df["ctlog_latest"].apply(
        lambda x: x.weekday() if isinstance(x, date) else None
    )

# set data type of the day of week columns to int
df["whois_created_dayofweek"] =
df["whois_created_dayofweek"].astype("int64")
df["ctlog_earliest_dayofweek"] =
df["ctlog_earliest_dayofweek"].astype("int64")
df["ctlog_latest_dayofweek"] = df["ctlog_latest_dayofweek"].astype("int64")

# domain to cert delta
df["domain_to_earliest_cert_delta"] = df.apply(
        lambda x: utils.get_days_delta(
            x["ctlog_earliest"],
            x["whois_created"]
            if x["whois_created"] and x["ctlog_earliest"]
```

```
            else None,
        ),
        axis=1,
    )

# set the data type of the domain_to_cert_delta column to float
df["domain_to_earliest_cert_delta"] =
df["domain_to_earliest_cert_delta"].astype("float64")

# domain to latest cert delta
df["domain_to_latest_cert_delta"] = df.apply(
        lambda x: utils.get_days_delta(
            x["ctlog_latest"],
            x["whois_created"]
            if x["whois_created"] and x["ctlog_latest"]
            else None,
        ),
        axis=1,
    )

# set the data type of the domain_to_cert_delta column to float
df["domain_to_latest_cert_delta"] =
df["domain_to_latest_cert_delta"].astype("float64")

# drop columns we imported that we will never use
df = df.drop(columns=["url", "verification_time", "dateadded",
"soa_timestamp","domain_to_cert_delta"])

click.echo(df.head())
click.echo(df.describe(include='all'))
click.echo(df.dtypes)
                            domain  malicious       whois_created
0  i-db5p-cor001.api.p001.1drv.com      False 2013-08-05 18:33:50  \
4       soundcloud-pax.pandora.com      False 1993-12-28 05:00:00
5         joolcomercializadora.com       True 2023-05-22 14:53:50
6        createpdf-asr.acrobat.com      False 1999-03-16 05:00:00
8                          popt.in      False 2016-05-14 16:58:55

       ctlog_earliest        ctlog_latest  ctlog_wildcard
0 2022-01-24 20:01:58 2023-06-08 20:46:06            True  \
4 2022-05-19 00:00:00 2023-06-19 23:59:59            True
5 2022-04-06 22:23:24 2023-09-22 23:59:59           False
6 2022-09-09 00:00:00 2023-10-10 23:59:59            True
8 2023-01-07 20:36:15 2023-08-15 04:16:52           False

   whois_created_dayofweek  ctlog_earliest_dayofweek
ctlog_latest_dayofweek
0                        0                          0
3  \
```

```
4                        1                          3
0
5                        0                          2
4
6                        1                          4
1
8                        5                          5
1

   domain_to_earliest_cert_delta  domain_to_latest_cert_delta
0                        -3095.0                      -3595.0
4                       -10369.0                     -10766.0
5                          410.0                       -124.0
6                        -8578.0                      -8975.0
8                        -2430.0                      -2649.0
                  domain malicious                 whois_created
count              21549     21549                         21549  \
unique             21536         2                           NaN
top     www.mediafire.com     False                          NaN
freq                   2     11739                           NaN
mean                 NaN       NaN  2012-10-03 12:56:32.335050496
min                  NaN       NaN            1986-01-09 00:00:00
25%                  NaN       NaN            2003-05-25 13:35:05
50%                  NaN       NaN            2015-05-07 23:56:05
75%                  NaN       NaN            2023-03-20 15:03:16
max                  NaN       NaN            2023-07-03 08:21:24
std                  NaN       NaN                           NaN

                        ctlog_earliest                  ctlog_latest
count                            21549                         21549  \
unique                             NaN                           NaN
top                                NaN                           NaN
freq                               NaN                           NaN
mean    2022-09-26 15:45:50.943570432  2023-08-14 17:49:06.400900352
min               2021-11-30 05:24:28            2023-01-01 18:42:11
25%               2022-06-24 13:47:12            2023-07-02 08:11:07
50%               2022-10-18 21:00:14            2023-08-21 21:40:11
75%               2022-12-14 00:00:00            2023-09-21 19:41:38
max               2023-06-28 04:36:22            2023-12-31 23:59:59
std                                NaN                           NaN

       ctlog_wildcard  whois_created_dayofweek  ctlog_earliest_dayofweek
count           21549             21549.000000              21549.000000  \
unique              2                      NaN                       NaN
top             False                      NaN                       NaN
freq            13032                      NaN                       NaN
mean              NaN                 2.332823                  2.399462
min               NaN                 0.000000                  0.000000
25%               NaN                 1.000000                  1.000000
```

```
50%              NaN                2.000000                   2.000000
75%              NaN                4.000000                   4.000000
max              NaN                6.000000                   6.000000
std              NaN                1.775043                   1.897252

        ctlog_latest_dayofweek  domain_to_earliest_cert_delta
count             21549.000000                   21549.000000  \
unique                     NaN                            NaN
top                        NaN                            NaN
freq                       NaN                            NaN
mean                  2.873080                   -3645.602070
min                   0.000000                  -13445.000000
25%                   1.000000                   -7078.000000
50%                   3.000000                   -2637.000000
75%                   5.000000                      69.000000
max                   6.000000                     524.000000
std                   2.057394                    3790.677119

        domain_to_latest_cert_delta
count                  21549.000000
unique                          NaN
top                             NaN
freq                            NaN
mean                   -3967.678222
min                   -13798.000000
25%                    -7421.000000
50%                    -3009.000000
75%                     -144.000000
max                      135.000000
std                     3852.703681
domain                                string[python]
malicious                                       bool
whois_created                         datetime64[ns]
ctlog_earliest                        datetime64[ns]
ctlog_latest                          datetime64[ns]
ctlog_wildcard                                  bool
whois_created_dayofweek                        int64
ctlog_earliest_dayofweek                       int64
ctlog_latest_dayofweek                         int64
domain_to_earliest_cert_delta                float64
domain_to_latest_cert_delta                  float64
dtype: object
```

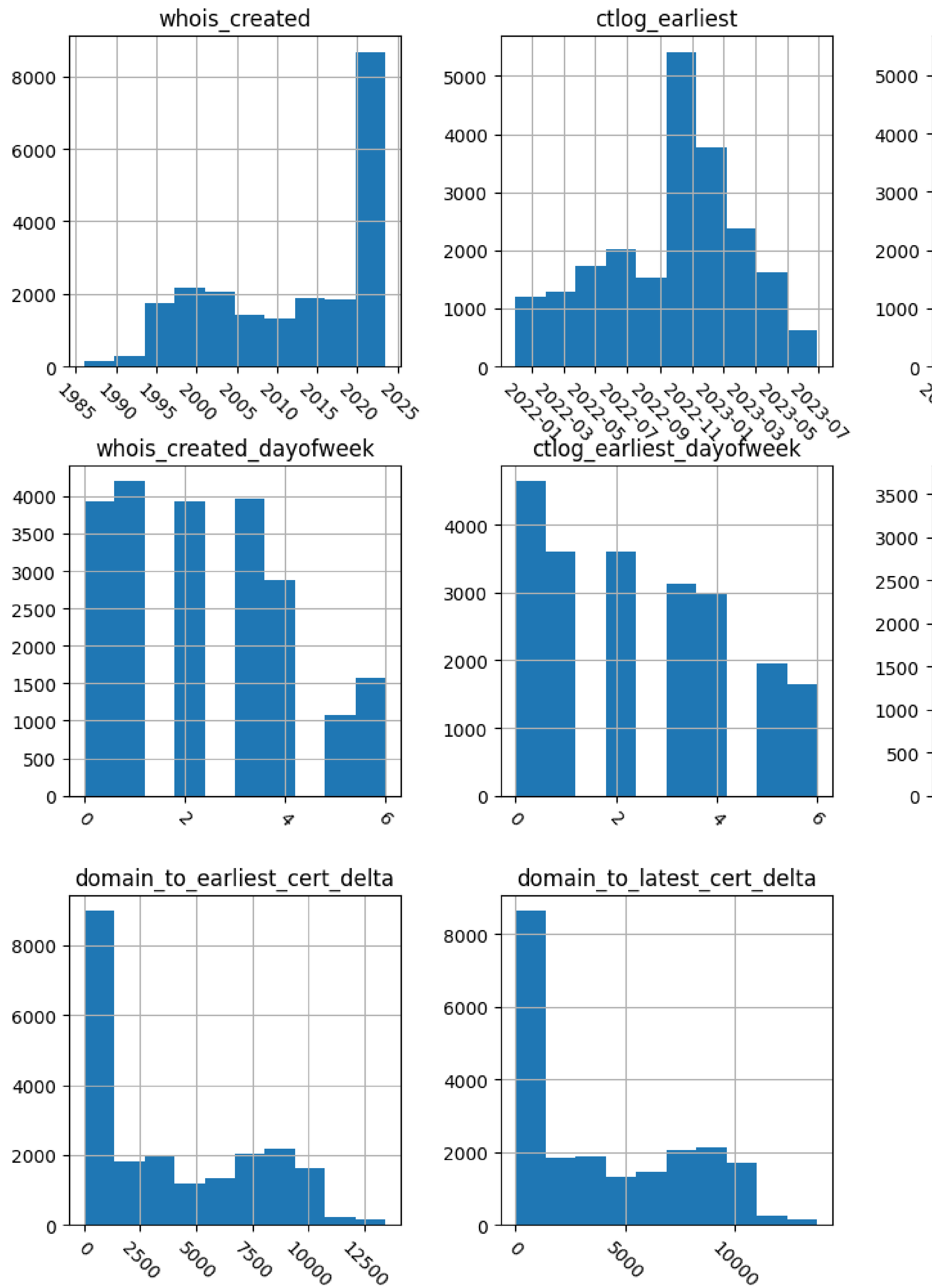
In [3]:

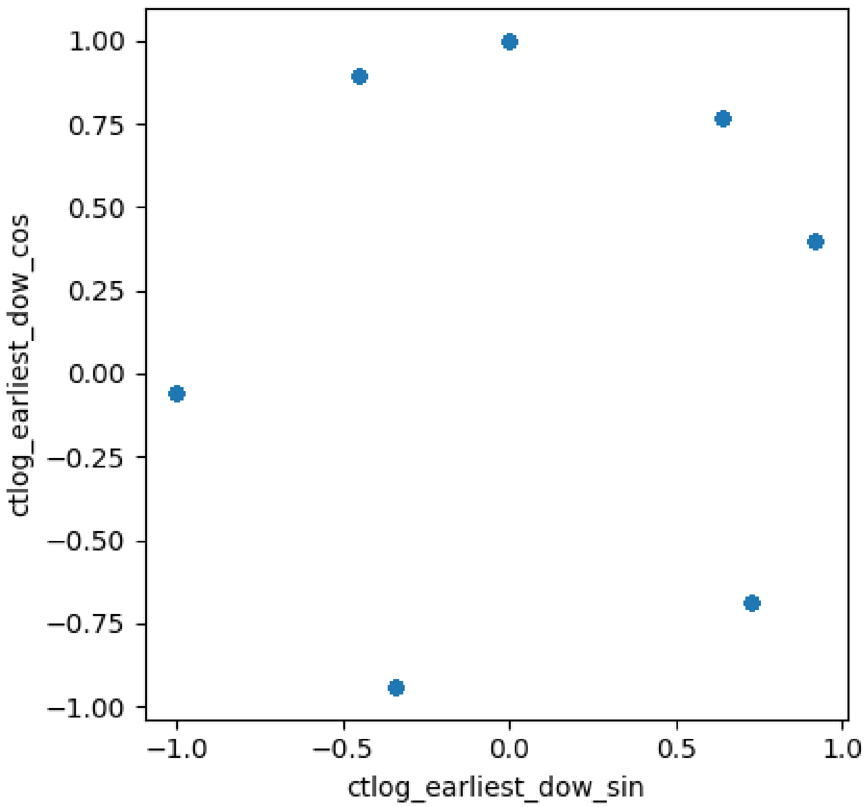
```
# absolute value of the domain_to_cert_delta
df["domain_to_earliest_cert_delta"] =
abs(df["domain_to_earliest_cert_delta"])
df["domain_to_latest_cert_delta"] = abs(df["domain_to_latest_cert_delta"])

df.hist(figsize=(12,12), xrot=-45)
```

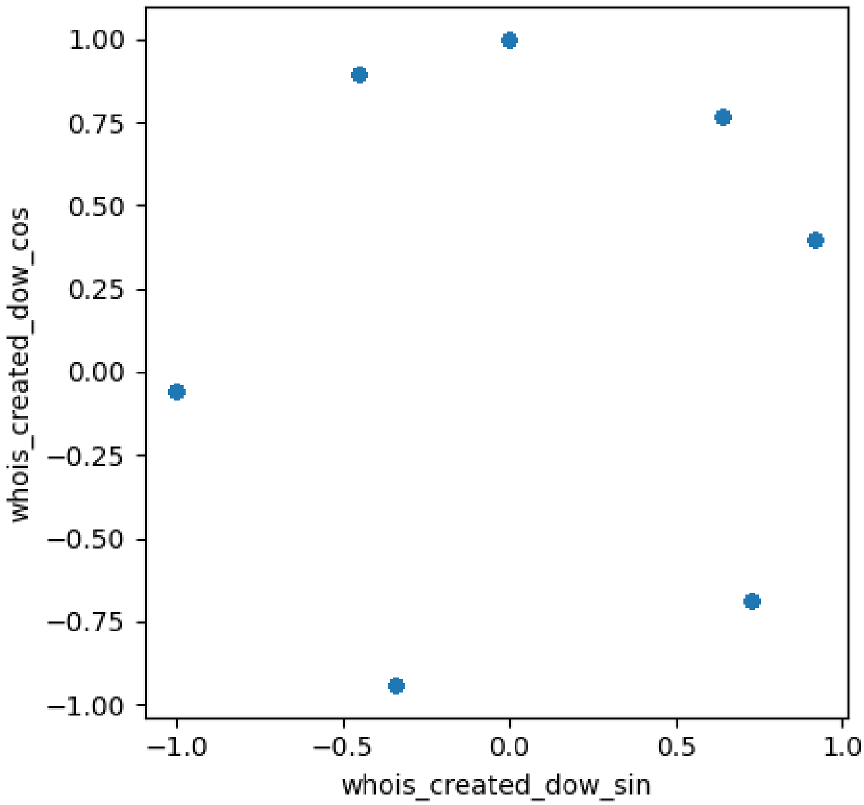
```
col_index = df.columns.get_loc("whois_created_dayofweek")
df["whois_created_dow_sin"] = df.apply(lambda row:
math.sin(360/7*(row[col_index])),axis=1)
df["whois_created_dow_cos"] = df.apply(lambda row:
math.cos(360/7*(row[col_index])),axis=1)

col_index = df.columns.get_loc("ctlog_earliest_dayofweek")
df["ctlog_earliest_dow_sin"] = df.apply(lambda row:
math.sin(360/7*(row[col_index])),axis=1)
df["ctlog_earliest_dow_cos"] = df.apply(lambda row:
math.cos(360/7*(row[col_index])),axis=1)

col_index = df.columns.get_loc("ctlog_latest_dayofweek")
df["ctlog_latest_dow_sin"] = df.apply(lambda row:
math.sin(360/7*(row[col_index])),axis=1)
df["ctlog_latest_dow_cos"] = df.apply(lambda row:
math.cos(360/7*(row[col_index])),axis=1)


df.plot.scatter(x="ctlog_earliest_dow_sin",
y="ctlog_earliest_dow_cos").set_aspect('equal')
df.plot.scatter(x="whois_created_dow_sin",
y="whois_created_dow_cos").set_aspect('equal')
df.plot.scatter(x="ctlog_latest_dow_sin",
y="ctlog_latest_dow_cos").set_aspect('equal')

"""
# add one hot encode ctlog_earliest_not_before_dayofweek
df = pd.get_dummies(
        df,
        columns=["ctlog_earliest_dayofweek"],
        prefix=["ctlog_earliest_dow"],
    )
# add one hot encode whois_created_dayofweek to columns
df = pd.get_dummies(
        df, columns=["whois_created_dayofweek"], prefix="whois_dow"
    )
    """

# Summary statistics
click.echo(df.describe(include='all'))
                   domain malicious                     whois_created
count               21549     21549                             21549  \
unique              21536         2                               NaN
top     www.mediafire.com     False                               NaN
freq                    2     11739                               NaN
mean                  NaN       NaN  2012-10-03 12:56:32.335050496
min                   NaN       NaN            1986-01-09 00:00:00
```

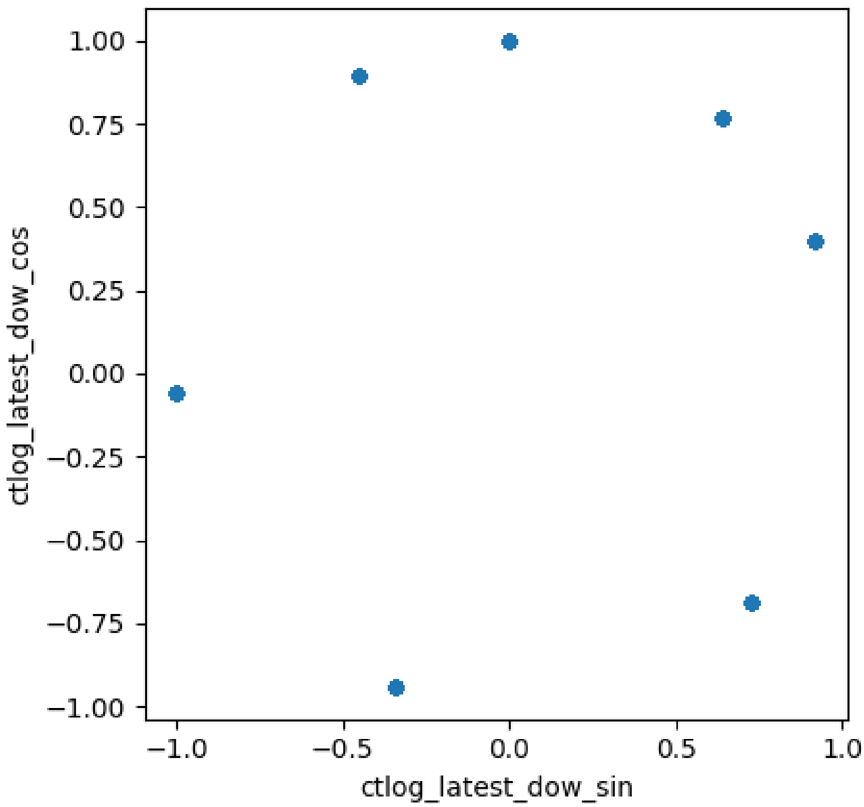
```
25%                  NaN       NaN             2003-05-25 13:35:05
50%                  NaN       NaN             2015-05-07 23:56:05
75%                  NaN       NaN             2023-03-20 15:03:16
max                  NaN       NaN             2023-07-03 08:21:24
std                  NaN       NaN                             NaN

                       ctlog_earliest                   ctlog_latest
count                           21549                          21549  \
unique                            NaN                            NaN
top                               NaN                            NaN
freq                              NaN                            NaN
mean    2022-09-26 15:45:50.943570432  2023-08-14 17:49:06.400900352
min               2021-11-30 05:24:28            2023-01-01 18:42:11
25%               2022-06-24 13:47:12            2023-07-02 08:11:07
50%               2022-10-18 21:00:14            2023-08-21 21:40:11
75%               2022-12-14 00:00:00            2023-09-21 19:41:38
max               2023-06-28 04:36:22            2023-12-31 23:59:59
std                               NaN                            NaN

       ctlog_wildcard  whois_created_dayofweek  ctlog_earliest_dayofweek
count           21549             21549.000000              21549.000000  \
unique              2                      NaN                       NaN
top             False                      NaN                       NaN
freq            13032                      NaN                       NaN
mean              NaN                 2.332823                  2.399462
min               NaN                 0.000000                  0.000000
25%               NaN                 1.000000                  1.000000
50%               NaN                 2.000000                  2.000000
75%               NaN                 4.000000                  4.000000
max               NaN                 6.000000                  6.000000
std               NaN                 1.775043                  1.897252

        ctlog_latest_dayofweek  domain_to_earliest_cert_delta
count             21549.000000                   21549.000000  \
unique                     NaN                            NaN
top                        NaN                            NaN
freq                       NaN                            NaN
mean                  2.873080                    3742.948397
min                   0.000000                       0.000000
25%                   1.000000                     181.000000
50%                   3.000000                    2637.000000
75%                   5.000000                    7078.000000
max                   6.000000                   13445.000000
std                   2.057394                    3694.584062

        domain_to_latest_cert_delta  whois_created_dow_sin
count                  21549.000000           21549.000000  \
unique                          NaN                    NaN
top                             NaN                    NaN
```

```
freq                           NaN                      NaN
mean                   3969.491206                 0.140419
min                       0.000000                -0.998199
25%                     144.000000                -0.340712
50%                    3009.000000                 0.000000
75%                    7421.000000                 0.728010
max                   13798.000000                 0.918032
std                    3850.835626                 0.659922

        whois_created_dow_cos  ctlog_earliest_dow_sin
ctlog_earliest_dow_cos
count            21549.000000            21549.000000
21549.000000  \
unique                    NaN                     NaN
NaN
top                       NaN                     NaN
NaN
freq                      NaN                     NaN
NaN
mean                 0.054288                0.095357
0.161451
min                 -0.940168               -0.998199               -
0.940168
25%                 -0.685567               -0.340712               -
0.685567
50%                  0.396506                0.000000
0.396506
75%                  0.767830                0.728010
0.892589
max                  1.000000                0.918032
1.000000
std                  0.736128                0.651782
0.734891

        ctlog_latest_dow_sin  ctlog_latest_dow_cos
count           21549.000000          21549.000000
unique                   NaN                   NaN
top                      NaN                   NaN
freq                     NaN                   NaN
mean                0.096253              0.255578
min                -0.998199             -0.940168
25%                -0.450871             -0.685567
50%                 0.000000              0.396506
75%                 0.728010              0.892589
max                 0.918032              1.000000
std                 0.651597              0.707728
```


whois_created
8000
6000
4000
2000
0
1985
1990
1995
2000
2005
2010
2015
2020
2025
ctlog_earliest
5000
4000
3000
2000
1000
0
2022-01
2022-03
2022-05
2022-07
2022-09
2022-11
2023-01
2023-03
2023-05
2023-07
5000
4000
3000
2000
1000
0
whois_created_dayofweek
4000
3500
3000
2500
2000
1500
1000
500
0
0
2
4
6
ctlog_earliest_dayofweek
4000
3000
2000
1000
0
0
2
4
6
3500
3000
2500
2000
1500
1000
500
0
domain_to_earliest_cert_delta
8000
6000
4000
2000
0
0
2500
5000
7500
10000
12500
domain_to_latest_cert_delta
8000
6000
4000
2000
0
0
5000
10000


1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
ctlog_earliest_dow_cos
−1.0
−0.5
0.0
0.5
1.0
ctlog_earliest_dow_sin

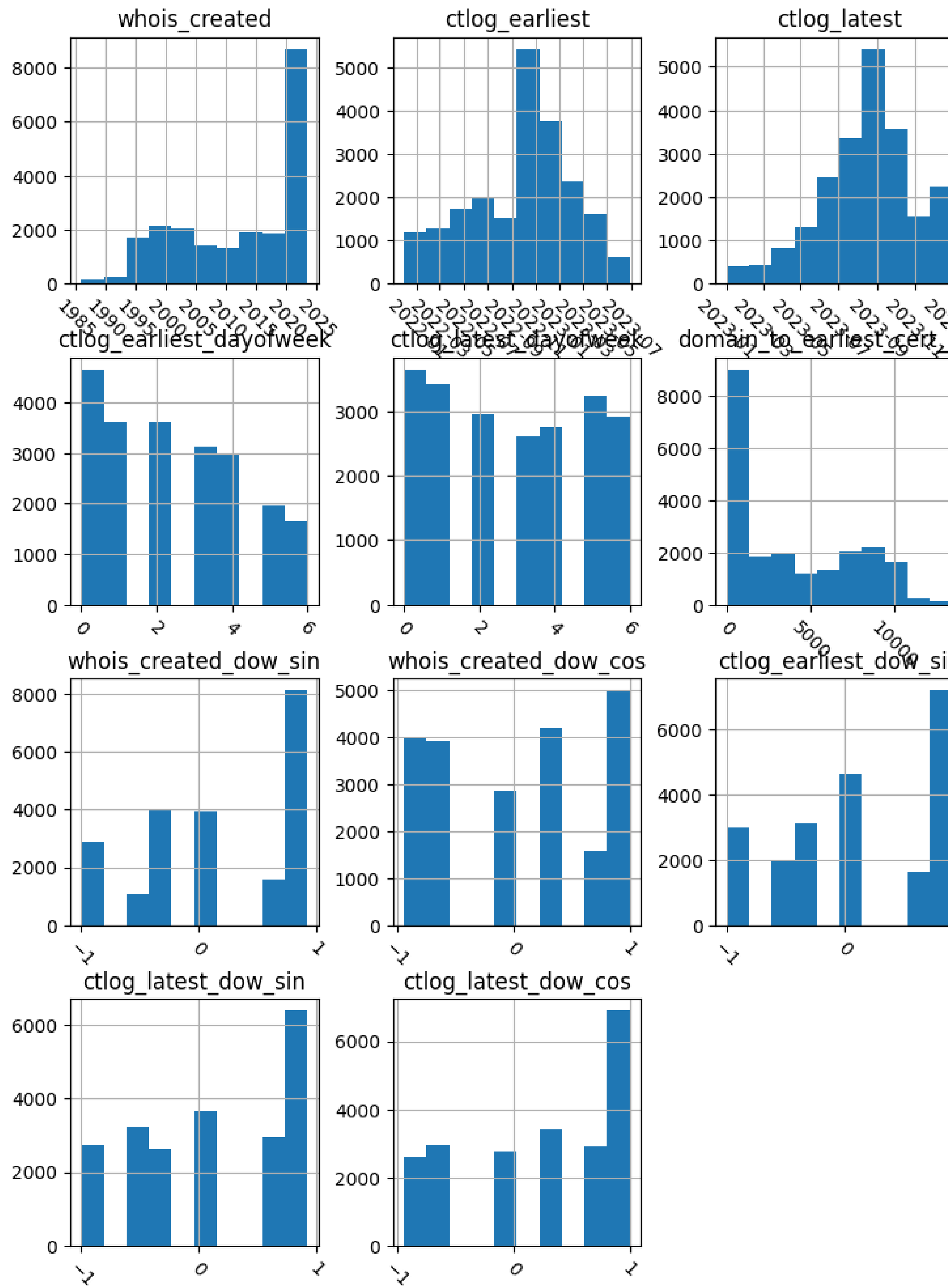

1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
whois_created_dow_cos
−1.0
−0.5
0.0
0.5
1.0
whois_created_dow_sin



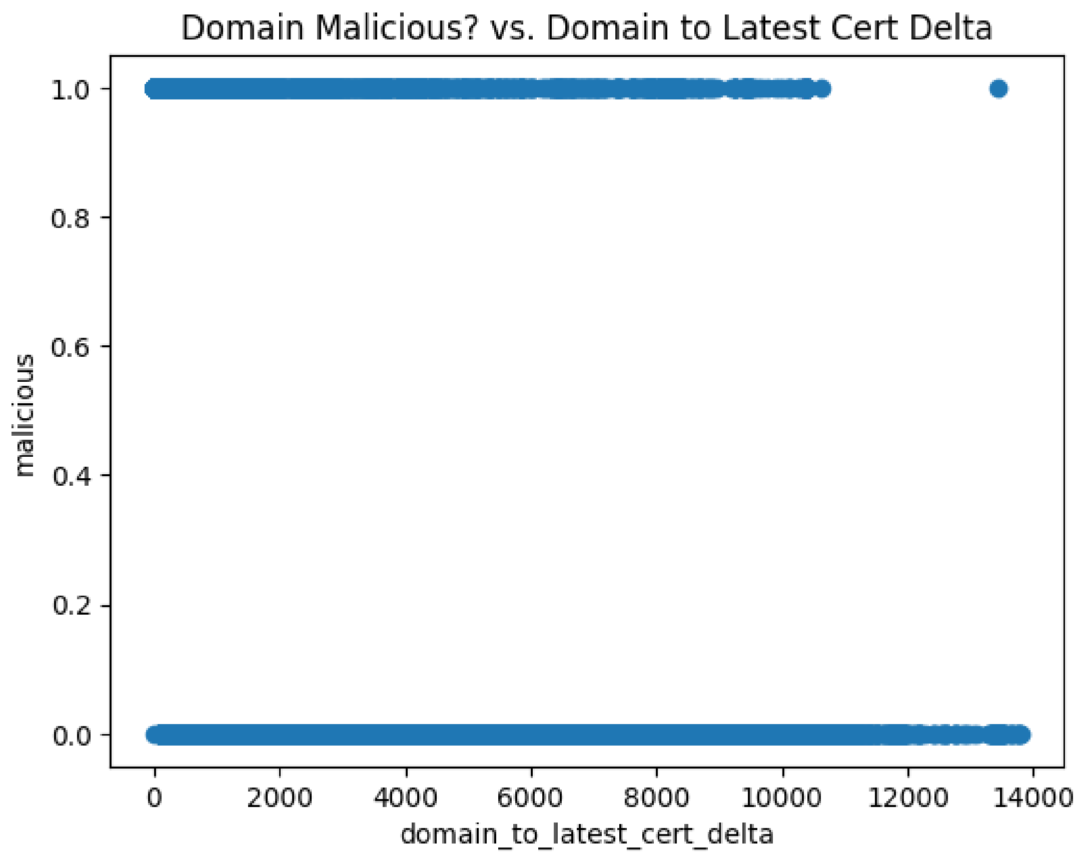
In [4]:

```
# Plot histograms
df.hist(figsize=(12,12), xrot=-45)

# Create a scatter plot of the data, showing malicious and benign domains
# plot the data as a scatter plot
plt.scatter(df["domain_to_earliest_cert_delta"], df["malicious"])
plt.xlabel("domain_to_earliest_cert_delta")
plt.ylabel("malicious")
plt.title("Domain Malicious? vs. Domain to Earliest Cert Delta")
plt.show()
# and for the latest
plt.scatter(df["domain_to_latest_cert_delta"], df["malicious"])
plt.xlabel("domain_to_latest_cert_delta")
plt.ylabel("malicious")
plt.title("Domain Malicious? vs. Domain to Latest Cert Delta")
plt.show()



click.echo(df.head())

# print a count of malicious and benign values
click.echo(df["malicious"].value_counts())
# deduplicate any rows, there shouldn't be any, but just in case
df = df.drop_duplicates()
click.echo(df["malicious"].value_counts())
```

```
click.echo(df.head())

X = df.drop(["malicious","domain", "ctlog_earliest", "ctlog_latest",
"whois_created", "whois_created_dayofweek", "ctlog_earliest_dayofweek"],
axis=1)
X = df.filter(combo_features)
y = df.filter(["malicious"])


click.echo(X.describe())
```



whois_created
ctlog_earliest
ctlog_latest
ctlog_earliest_dayofweek
ctlog_latest_dayofweek
domain_to_earliest_cert_
whois_created_dow_sin
whois_created_dow_cos
ctlog_earliest_dow_si
ctlog_latest_dow_sin
ctlog_latest_dow_cos




```
                              domain  malicious       whois_created
0  i-db5p-cor001.api.p001.1drv.com      False 2013-08-05 18:33:50  \
4       soundcloud-pax.pandora.com      False 1993-12-28 05:00:00
5         joolcomercializadora.com       True 2023-05-22 14:53:50
6        createpdf-asr.acrobat.com      False 1999-03-16 05:00:00
8                          popt.in      False 2016-05-14 16:58:55

       ctlog_earliest        ctlog_latest  ctlog_wildcard  \
0 2022-01-24 20:01:58 2023-06-08 20:46:06            True
4 2022-05-19 00:00:00 2023-06-19 23:59:59            True
5 2022-04-06 22:23:24 2023-09-22 23:59:59           False
6 2022-09-09 00:00:00 2023-10-10 23:59:59            True
8 2023-01-07 20:36:15 2023-08-15 04:16:52           False

   whois_created_dayofweek  ctlog_earliest_dayofweek
ctlog_latest_dayofweek
0                        0                          0
3  \
4                        1                          3
0
5                        0                          2
4
6                        1                          4
1
8                        5                          5
1
```

```
   domain_to_earliest_cert_delta  domain_to_latest_cert_delta
0                         3095.0                       3595.0  \
4                        10369.0                      10766.0
5                          410.0                        124.0
6                         8578.0                       8975.0
8                         2430.0                       2649.0

   whois_created_dow_sin  whois_created_dow_cos  ctlog_earliest_dow_sin
0               0.000000               1.000000                0.000000  \
4               0.918032               0.396506               -0.340712
5               0.000000               1.000000                0.728010
6               0.918032               0.396506               -0.998199
8              -0.450871               0.892589               -0.450871

   ctlog_earliest_dow_cos  ctlog_latest_dow_sin  ctlog_latest_dow_cos
0                1.000000             -0.340712             -0.940168
4               -0.940168              0.000000              1.000000
5               -0.685567             -0.998199             -0.059997
6               -0.059997              0.918032              0.396506
8                0.892589              0.918032              0.396506
malicious
False    11739
True      9810
Name: count, dtype: int64
malicious
False    11739
True      9810
Name: count, dtype: int64
                            domain  malicious       whois_created
0  i-db5p-cor001.api.p001.1drv.com      False 2013-08-05 18:33:50  \
4       soundcloud-pax.pandora.com      False 1993-12-28 05:00:00
5         joolcomercializadora.com       True 2023-05-22 14:53:50
6        createpdf-asr.acrobat.com      False 1999-03-16 05:00:00
8                          popt.in      False 2016-05-14 16:58:55

       ctlog_earliest        ctlog_latest  ctlog_wildcard
0 2022-01-24 20:01:58 2023-06-08 20:46:06            True  \
4 2022-05-19 00:00:00 2023-06-19 23:59:59            True
5 2022-04-06 22:23:24 2023-09-22 23:59:59           False
6 2022-09-09 00:00:00 2023-10-10 23:59:59            True
8 2023-01-07 20:36:15 2023-08-15 04:16:52           False

   whois_created_dayofweek  ctlog_earliest_dayofweek
ctlog_latest_dayofweek
0                        0                         0
3  \
4                        1                         3
0
```

```
5                        0                          2
4
6                        1                          4
1
8                        5                          5
1

   domain_to_earliest_cert_delta  domain_to_latest_cert_delta
0                         3095.0                       3595.0  \
4                        10369.0                      10766.0
5                          410.0                        124.0
6                         8578.0                       8975.0
8                         2430.0                       2649.0

   whois_created_dow_sin  whois_created_dow_cos  ctlog_earliest_dow_sin
0               0.000000               1.000000                0.000000  \
4               0.918032               0.396506               -0.340712
5               0.000000               1.000000                0.728010
6               0.918032               0.396506               -0.998199
8              -0.450871               0.892589               -0.450871

   ctlog_earliest_dow_cos  ctlog_latest_dow_sin  ctlog_latest_dow_cos
0                1.000000             -0.340712             -0.940168
4               -0.940168              0.000000              1.000000
5               -0.685567             -0.998199             -0.059997
6               -0.059997              0.918032              0.396506
8                0.892589              0.918032              0.396506
       domain_to_earliest_cert_delta  domain_to_latest_cert_delta
count                   21549.000000                 21549.000000  \
mean                     3742.948397                  3969.491206
std                      3694.584062                  3850.835626
min                         0.000000                     0.000000
25%                       181.000000                   144.000000
50%                      2637.000000                  3009.000000
75%                      7078.000000                  7421.000000
max                     13445.000000                 13798.000000

       ctlog_earliest_dow_sin  ctlog_earliest_dow_cos  ctlog_latest_dow_sin
count            21549.000000            21549.000000          21549.000000
\
mean                 0.095357                0.161451              0.096253
std                  0.651782                0.734891              0.651597
min                 -0.998199               -0.940168             -0.998199
25%                 -0.340712               -0.685567             -0.450871
50%                  0.000000                0.396506              0.000000
75%                  0.728010                0.892589              0.728010
max                  0.918032                1.000000              0.918032

       ctlog_latest_dow_cos
```

```
count          21549.000000
mean               0.255578
std                0.707728
min               -0.940168
25%               -0.685567
50%                0.396506
75%                0.892589
max                1.000000
```

In [5]:

```
# convert y (malicious) to 1/0 int
y = y.astype('int')
# convert X["ctlog_wildcard"]  to 1/0 int
if "ctlog_wildcard" in X.columns:
    X["ctlog_wildcard"] = X["ctlog_wildcard"].astype('int')

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

X_train = sm.add_constant(X_train)
X_test = sm.add_constant(X_test)

smfit = sm.Logit(y_train,X_train).fit()

smfit.summary()
Optimization terminated successfully.
         Current function value: 0.285414
         Iterations 7
```

Out[5]:

Logit Regression Results

Dep. Variable:	malicious	**No. Observations:**	17239
Model:	Logit	**Df Residuals:**	17231
Method:	MLE	**Df Model:**	7
Date:	Tue, 08 Aug 2023	**Pseudo R-squ.:**	0.5860
Time:	19:09:37	**Log-Likelihood:**	-4920.2
converged:	True	**LL-Null:**	-11885.
Covariance Type:	nonrobust	**LLR p-value:**	0.000

	coef	**std err**	**z**	**P>|z|**	**[0.025**	**0.975]**
const	3.0688	0.055	55.885	0.000	2.961	3.176
domain_to_earliest_cert_delta	0.0077	0.000	39.010	0.000	0.007	0.008
domain_to_latest_cert_delta	-0.0082	0.000	-41.482	0.000	-0.009	-0.008
ctlog_earliest_dow_sin	0.1165	0.040	2.893	0.004	0.038	0.195
ctlog_earliest_dow_cos	-0.3294	0.036	-9.257	0.000	-0.399	-0.260
ctlog_latest_dow_sin	-0.1553	0.040	-3.890	0.000	-0.234	-0.077
ctlog_latest_dow_cos	0.2591	0.038	6.898	0.000	0.185	0.333
ctlog_wildcard	-0.1649	0.058	-2.843	0.004	-0.279	-0.051

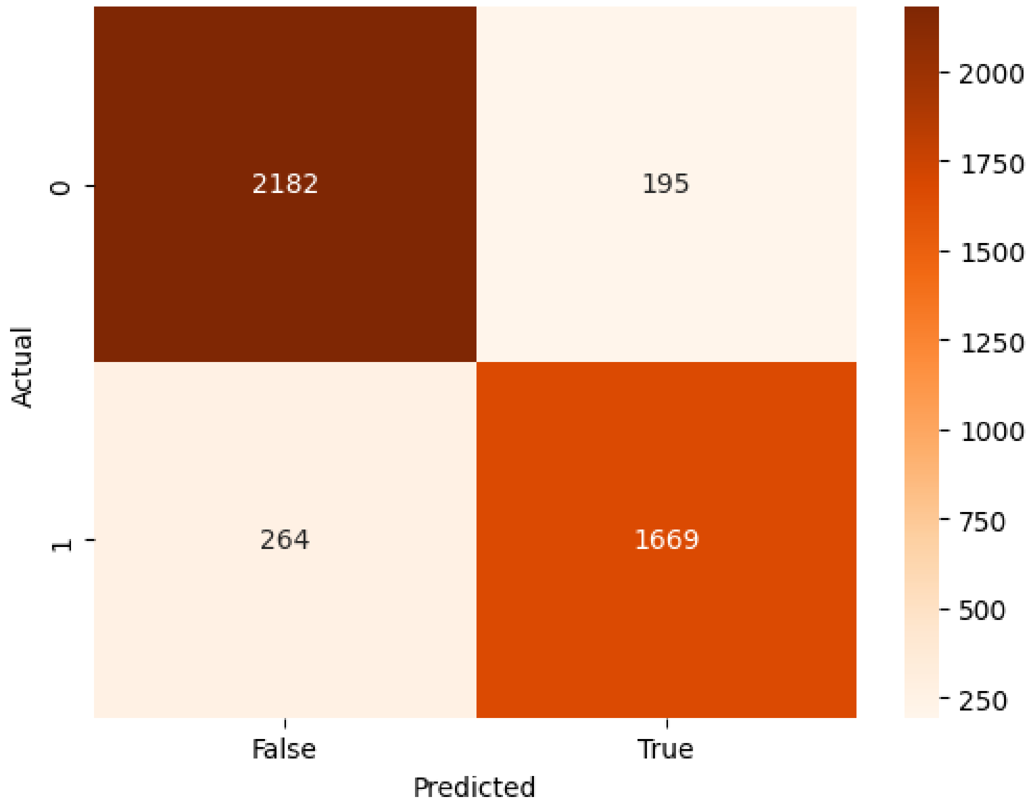
In [6]:

```
# Predict the malicious column using the test data
#add the incepts
```

```
y_predicted = smfit.predict(X_test)

# Present the results in a confusion matrix
confusion_matrix = confusion_matrix(y_test, y_predicted.round())
click.echo(confusion_matrix)

click.echo("Classification report:")
click.echo(classification_report(y_test, y_predicted.round()))

# Heatmap of confusion matrix
y_predicted

threshold = 0.5
y_predicted = [y > threshold for y in y_predicted]
data = {'Actual':    y_test.values.flatten(),
        'Predicted': y_predicted
        }

# Generate a confusion matrix and heatmap to evaluate the Type I and Type
II errors/ FP/FN etc.
df = pd.DataFrame(data, columns=['Actual','Predicted'])
confusion_matrix = pd.crosstab(df['Actual'], df['Predicted'],
rownames=['Actual'], colnames=['Predicted'])
fig = sns.heatmap(confusion_matrix, annot=True, cmap='Oranges', fmt='g')
fig
[[2182  195]
 [ 264 1669]]
Classification report:
              precision    recall  f1-score   support

           0       0.89      0.92      0.90      2377
           1       0.90      0.86      0.88      1933

    accuracy                           0.89      4310
   macro avg       0.89      0.89      0.89      4310
weighted avg       0.89      0.89      0.89      4310
```

Out[6]:

```
<Axes: xlabel='Predicted', ylabel='Actual'>
```



2182
195
264
1669
0
1
Actual
False
True
Predicted
2000
1750
1500
1250
1000
750
500
250

VII. Feature Set F

In [1]:

```
import click
import pandas as pd
import glob
import os
#import sklearn
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report
from sklearn.preprocessing import StandardScaler
from scipy import stats
from sklearn.linear_model import LogisticRegression
import statsmodels.api as sm
import matplotlib.pyplot as plt
from datetime import datetime, date
# qqplot of the data
import seaborn as sns
import math
import numpy as np
import utils

combo_features =     ['domain_to_earliest_cert_delta',
'whois_created_dow_sin', 'whois_created_dow_cos']

path = "../data/"
filename = None

epoch = datetime(1970,1,1)
# open the training data file, from ../data/ for the most recent merged
training data file saved by prepare-training-data-parallel.py
full_path = path + "merged_20230705-104357_training_adorned-engineered.csv"
# get latest file matching the glob
if not filename:
    filename = max(glob.glob(full_path), key=os.path.getctime)
    click.echo(f"Using {filename} as training data")
    df = pd.read_csv(filename, sep=",")

# shuffle the rows
df = df.sample(frac=1, random_state=42).reset_index(drop=True)

click.echo(df.shape)
click.echo(df.columns)

""" # From prepare-training-data-parallel.py:
# Columns:
url
verification_time
domain
```

```
malicious
dateadded
whois_created
soa_timestamp
ctlog_earliest
ctlog_latest
ctlog_wildcard
whois_created_dayofweek,
ctlog_earliest_dayofweek,
domain_to_cert_delta
"""


# Data cleaning - there may be some epoch values in the whois_created
#  & ct_log_earliest columns, so we need to remove those rows

# convert the whois_created and ctlog_earliest, ctlog_latest columns to
datetime

df = df.loc[df["whois_created"] != ""]
df = df.loc[df["ctlog_earliest"] != ""]
df = df.loc[df["ctlog_latest"] != ""]

# some dates are have nanoseconds in them, so we need to remove the
nanosecond part
df["whois_created"] = df["whois_created"].str.split(".", n = 1, expand =
True)[0]
# some dates has additional timezone information, so we need to remove that
df["whois_created"] = df["whois_created"].apply(lambda x: " ".join(
str(x).split(" ")[:2]))


# convert the whois_created and ct_log_earliest columns to datetime
df = df.astype({"whois_created": 'datetime64[ns]', "ctlog_earliest":
'datetime64[ns]', "ctlog_latest": 'datetime64[ns]'})
df = df.loc[df["whois_created"].ne(pd.Timestamp('1970-01-01 00:00:00'))]
df = df.loc[df["ctlog_earliest"].ne(pd.Timestamp('1970-01-01 00:00:00'))]
df = df.loc[df["ctlog_latest"].ne(pd.Timestamp('1970-01-01 00:00:00'))]


# malicious is a bool
df['malicious'] = df['malicious'].astype('bool')
df['domain'] = df['domain'].astype('string')
```

```
Using ../data/merged_20230705-104357_training_adorned-engineered.csv as
training data
(35438, 13)
Index(['url', 'verification_time', 'domain', 'malicious', 'dateadded',
       'whois_created', 'soa_timestamp', 'ctlog_earliest', 'ctlog_latest',
```

```
       'ctlog_wildcard', 'whois_created_dayofweek',
'ctlog_earliest_dayofweek',
       'domain_to_cert_delta'],
      dtype='object')
```

In [2]:

```
#########
# Feature engineering
#########

# remove any rows where the whois_created or ctlog_earliest columns are
null
df = df.loc[df["whois_created"].notnull()]
df = df.loc[df["ctlog_earliest"].notnull()]

# if the data is missing the "whois_created_dayofweek" or
"ctlog_earliest_dayofweek" columns, then add them
# whois created day of the week 0 = Monday, 6 = Sunday
df["whois_created_dayofweek"] = df["whois_created"].apply(
        lambda x: x.weekday() if isinstance(x, date) else None
    )

# cert valid day of the week 0 = Monday, 6 = Sunday
df["ctlog_earliest_dayofweek"] = df["ctlog_earliest"].apply(
        lambda x: x.weekday() if isinstance(x, date) else None
    )

df["ctlog_latest_dayofweek"] = df["ctlog_latest"].apply(
        lambda x: x.weekday() if isinstance(x, date) else None
    )

# set data type of the day of week columns to int
df["whois_created_dayofweek"] =
df["whois_created_dayofweek"].astype("int64")
df["ctlog_earliest_dayofweek"] =
df["ctlog_earliest_dayofweek"].astype("int64")
df["ctlog_latest_dayofweek"] = df["ctlog_latest_dayofweek"].astype("int64")

# domain to cert delta
df["domain_to_earliest_cert_delta"] = df.apply(
        lambda x: utils.get_days_delta(
            x["ctlog_earliest"],
            x["whois_created"]
            if x["whois_created"] and x["ctlog_earliest"]
            else None,
        ),
        axis=1,
    )

# set the data type of the domain_to_cert_delta column to float
```

```
df["domain_to_earliest_cert_delta"] =
df["domain_to_earliest_cert_delta"].astype("float64")

# domain to latest cert delta
df["domain_to_latest_cert_delta"] = df.apply(
        lambda x: utils.get_days_delta(
            x["ctlog_latest"],
            x["whois_created"]
            if x["whois_created"] and x["ctlog_latest"]
            else None,
        ),
        axis=1,
    )

# set the data type of the domain_to_cert_delta column to float
df["domain_to_latest_cert_delta"] =
df["domain_to_latest_cert_delta"].astype("float64")

# drop columns we imported that we will never use
df = df.drop(columns=["url", "verification_time", "dateadded",
"soa_timestamp","domain_to_cert_delta"])

click.echo(df.head())
click.echo(df.describe(include='all'))
click.echo(df.dtypes)
                            domain  malicious       whois_created
0  i-db5p-cor001.api.p001.1drv.com      False 2013-08-05 18:33:50  \
4       soundcloud-pax.pandora.com      False 1993-12-28 05:00:00
5         joolcomercializadora.com       True 2023-05-22 14:53:50
6        createpdf-asr.acrobat.com      False 1999-03-16 05:00:00
8                          popt.in      False 2016-05-14 16:58:55

       ctlog_earliest        ctlog_latest  ctlog_wildcard
0 2022-01-24 20:01:58 2023-06-08 20:46:06            True  \
4 2022-05-19 00:00:00 2023-06-19 23:59:59            True
5 2022-04-06 22:23:24 2023-09-22 23:59:59           False
6 2022-09-09 00:00:00 2023-10-10 23:59:59            True
8 2023-01-07 20:36:15 2023-08-15 04:16:52           False

   whois_created_dayofweek  ctlog_earliest_dayofweek
ctlog_latest_dayofweek
0                        0                          0
3  \
4                        1                          3
0
5                        0                          2
4
6                        1                          4
1
```

```
8                        5                          5
1

   domain_to_earliest_cert_delta  domain_to_latest_cert_delta
0                        -3095.0                      -3595.0
4                       -10369.0                     -10766.0
5                          410.0                       -124.0
6                        -8578.0                      -8975.0
8                        -2430.0                      -2649.0
                  domain malicious                  whois_created
count              21549     21549                          21549  \
unique             21536         2                            NaN
top     www.mediafire.com    False                            NaN
freq                   2     11739                            NaN
mean                 NaN       NaN  2012-10-03 12:56:32.335050496
min                  NaN       NaN            1986-01-09 00:00:00
25%                  NaN       NaN            2003-05-25 13:35:05
50%                  NaN       NaN            2015-05-07 23:56:05
75%                  NaN       NaN            2023-03-20 15:03:16
max                  NaN       NaN            2023-07-03 08:21:24
std                  NaN       NaN                            NaN

                        ctlog_earliest                   ctlog_latest
count                            21549                          21549  \
unique                             NaN                            NaN
top                                NaN                            NaN
freq                               NaN                            NaN
mean    2022-09-26 15:45:50.943570432  2023-08-14 17:49:06.400900352
min               2021-11-30 05:24:28            2023-01-01 18:42:11
25%               2022-06-24 13:47:12            2023-07-02 08:11:07
50%               2022-10-18 21:00:14            2023-08-21 21:40:11
75%               2022-12-14 00:00:00            2023-09-21 19:41:38
max               2023-06-28 04:36:22            2023-12-31 23:59:59
std                                NaN                            NaN

       ctlog_wildcard  whois_created_dayofweek  ctlog_earliest_dayofweek
count           21549             21549.000000              21549.000000  \
unique              2                      NaN                       NaN
top             False                      NaN                       NaN
freq            13032                      NaN                       NaN
mean              NaN                 2.332823                  2.399462
min               NaN                 0.000000                  0.000000
25%               NaN                 1.000000                  1.000000
50%               NaN                 2.000000                  2.000000
75%               NaN                 4.000000                  4.000000
max               NaN                 6.000000                  6.000000
std               NaN                 1.775043                  1.897252

        ctlog_latest_dayofweek  domain_to_earliest_cert_delta
```

```
count            21549.000000                21549.000000  \
unique                    NaN                         NaN
top                       NaN                         NaN
freq                      NaN                         NaN
mean                 2.873080                -3645.602070
min                  0.000000               -13445.000000
25%                  1.000000                -7078.000000
50%                  3.000000                -2637.000000
75%                  5.000000                   69.000000
max                  6.000000                  524.000000
std                  2.057394                 3790.677119

        domain_to_latest_cert_delta
count                  21549.000000
unique                          NaN
top                             NaN
freq                            NaN
mean                   -3967.678222
min                   -13798.000000
25%                    -7421.000000
50%                    -3009.000000
75%                     -144.000000
max                      135.000000
std                     3852.703681
domain                            string[python]
malicious                                   bool
whois_created                     datetime64[ns]
ctlog_earliest                    datetime64[ns]
ctlog_latest                      datetime64[ns]
ctlog_wildcard                              bool
whois_created_dayofweek                    int64
ctlog_earliest_dayofweek                   int64
ctlog_latest_dayofweek                     int64
domain_to_earliest_cert_delta            float64
domain_to_latest_cert_delta              float64
dtype: object
```

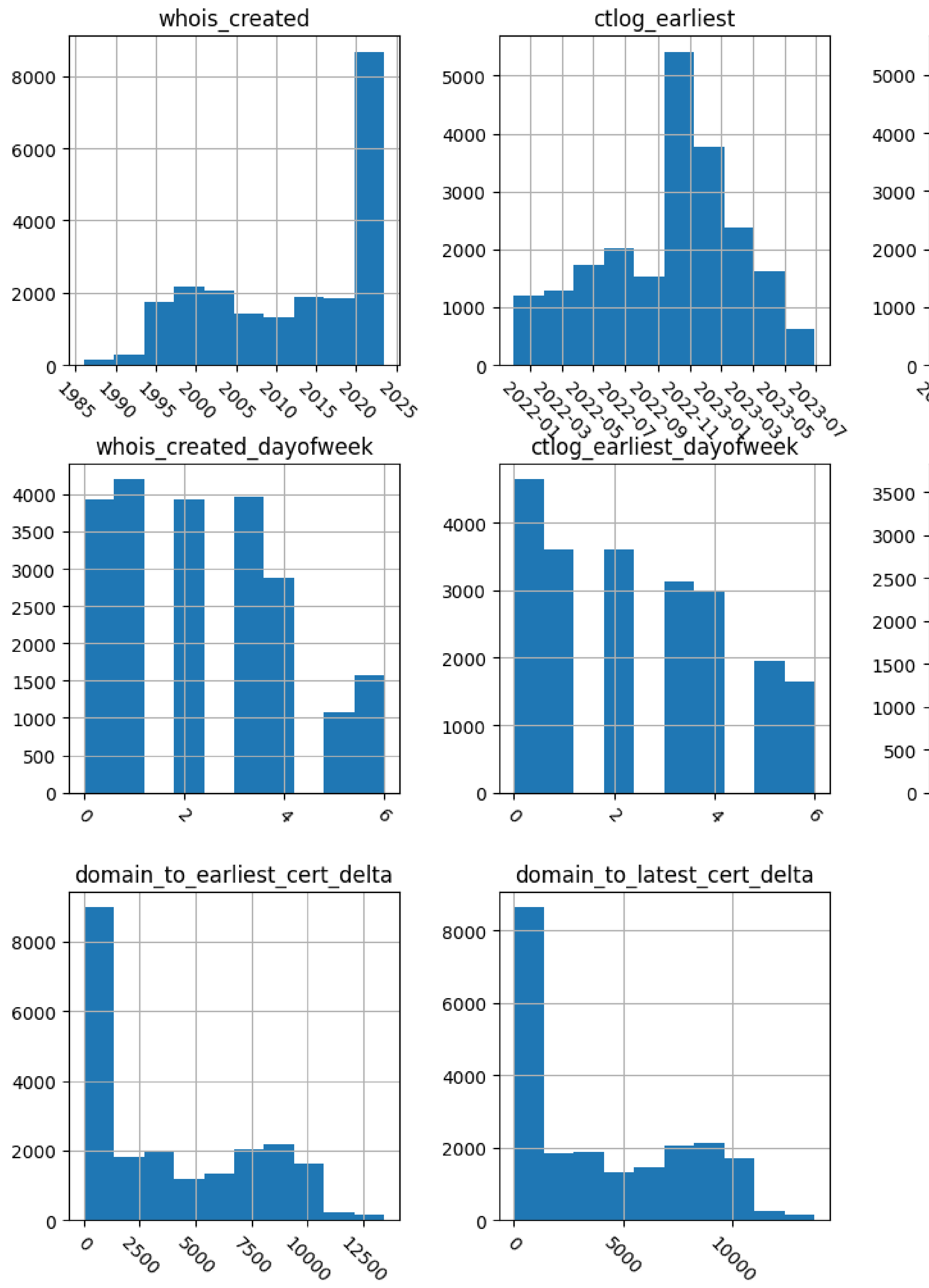
In [3]:

```
# absolute value of the domain_to_cert_delta
df["domain_to_earliest_cert_delta"] =
abs(df["domain_to_earliest_cert_delta"])
df["domain_to_latest_cert_delta"] = abs(df["domain_to_latest_cert_delta"])

df.hist(figsize=(12,12), xrot=-45)

col_index = df.columns.get_loc("whois_created_dayofweek")
df["whois_created_dow_sin"] = df.apply(lambda row:
math.sin(360/7*(row[col_index])),axis=1)
df["whois_created_dow_cos"] = df.apply(lambda row:
math.cos(360/7*(row[col_index])),axis=1)
```

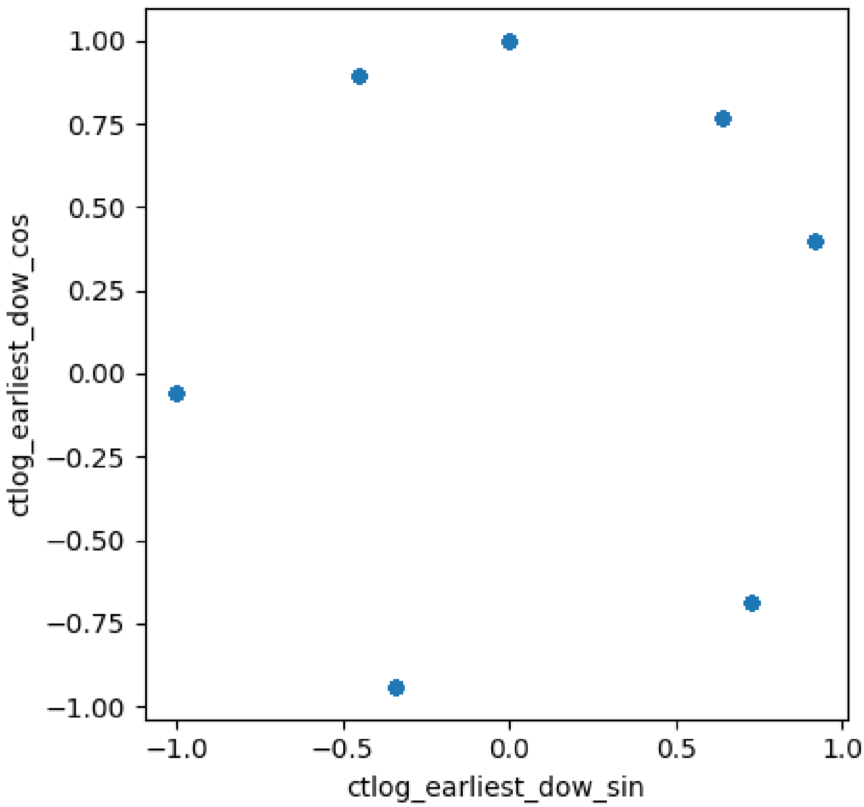
```
col_index = df.columns.get_loc("ctlog_earliest_dayofweek")
df["ctlog_earliest_dow_sin"] = df.apply(lambda row:
math.sin(360/7*(row[col_index])),axis=1)
df["ctlog_earliest_dow_cos"] = df.apply(lambda row:
math.cos(360/7*(row[col_index])),axis=1)

col_index = df.columns.get_loc("ctlog_latest_dayofweek")
df["ctlog_latest_dow_sin"] = df.apply(lambda row:
math.sin(360/7*(row[col_index])),axis=1)
df["ctlog_latest_dow_cos"] = df.apply(lambda row:
math.cos(360/7*(row[col_index])),axis=1)


df.plot.scatter(x="ctlog_earliest_dow_sin",
y="ctlog_earliest_dow_cos").set_aspect('equal')
df.plot.scatter(x="whois_created_dow_sin",
y="whois_created_dow_cos").set_aspect('equal')
df.plot.scatter(x="ctlog_latest_dow_sin",
y="ctlog_latest_dow_cos").set_aspect('equal')

"""
# add one hot encode ctlog_earliest_not_before_dayofweek
df = pd.get_dummies(
        df,
        columns=["ctlog_earliest_dayofweek"],
        prefix=["ctlog_earliest_dow"],
    )
# add one hot encode whois_created_dayofweek to columns
df = pd.get_dummies(
        df, columns=["whois_created_dayofweek"], prefix="whois_dow"
    )
    """

# Summary statistics
click.echo(df.describe(include='all'))
                   domain malicious                  whois_created
count               21549     21549                          21549  \
unique              21536         2                            NaN
top     www.mediafire.com     False                            NaN
freq                    2     11739                            NaN
mean                  NaN       NaN  2012-10-03 12:56:32.335050496
min                   NaN       NaN            1986-01-09 00:00:00
25%                   NaN       NaN            2003-05-25 13:35:05
50%                   NaN       NaN            2015-05-07 23:56:05
75%                   NaN       NaN            2023-03-20 15:03:16
max                   NaN       NaN            2023-07-03 08:21:24
std                   NaN       NaN                            NaN
```

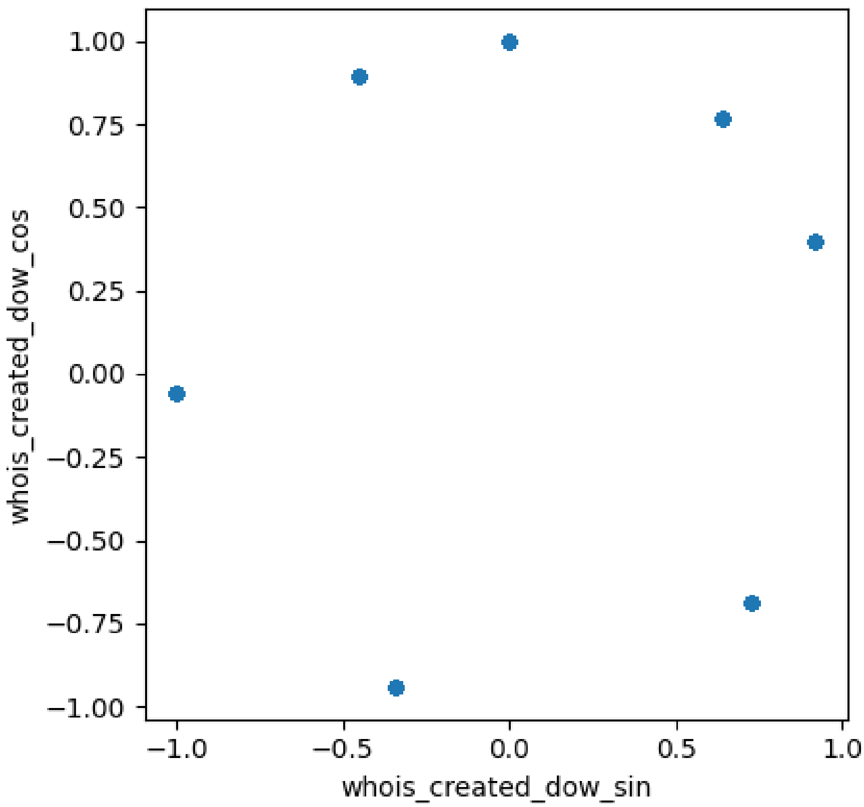
```
                    ctlog_earliest                ctlog_latest
count                        21549                       21549  \
unique                         NaN                         NaN
top                            NaN                         NaN
freq                           NaN                         NaN
mean    2022-09-26 15:45:50.943570432  2023-08-14 17:49:06.400900352
min            2021-11-30 05:24:28         2023-01-01 18:42:11
25%            2022-06-24 13:47:12         2023-07-02 08:11:07
50%            2022-10-18 21:00:14         2023-08-21 21:40:11
75%            2022-12-14 00:00:00         2023-09-21 19:41:38
max            2023-06-28 04:36:22         2023-12-31 23:59:59
std                            NaN                         NaN

       ctlog_wildcard  whois_created_dayofweek  ctlog_earliest_dayofweek
count           21549              21549.000000              21549.000000  \
unique              2                       NaN                       NaN
top             False                       NaN                       NaN
freq            13032                       NaN                       NaN
mean              NaN                  2.332823                  2.399462
min               NaN                  0.000000                  0.000000
25%               NaN                  1.000000                  1.000000
50%               NaN                  2.000000                  2.000000
75%               NaN                  4.000000                  4.000000
max               NaN                  6.000000                  6.000000
std               NaN                  1.775043                  1.897252

        ctlog_latest_dayofweek  domain_to_earliest_cert_delta
count             21549.000000                   21549.000000  \
unique                     NaN                            NaN
top                        NaN                            NaN
freq                       NaN                            NaN
mean                  2.873080                    3742.948397
min                   0.000000                       0.000000
25%                   1.000000                     181.000000
50%                   3.000000                    2637.000000
75%                   5.000000                    7078.000000
max                   6.000000                   13445.000000
std                   2.057394                    3694.584062

        domain_to_latest_cert_delta  whois_created_dow_sin
count                  21549.000000           21549.000000  \
unique                          NaN                    NaN
top                             NaN                    NaN
freq                            NaN                    NaN
mean                    3969.491206               0.140419
min                        0.000000              -0.998199
25%                      144.000000              -0.340712
50%                     3009.000000               0.000000
75%                     7421.000000               0.728010
```

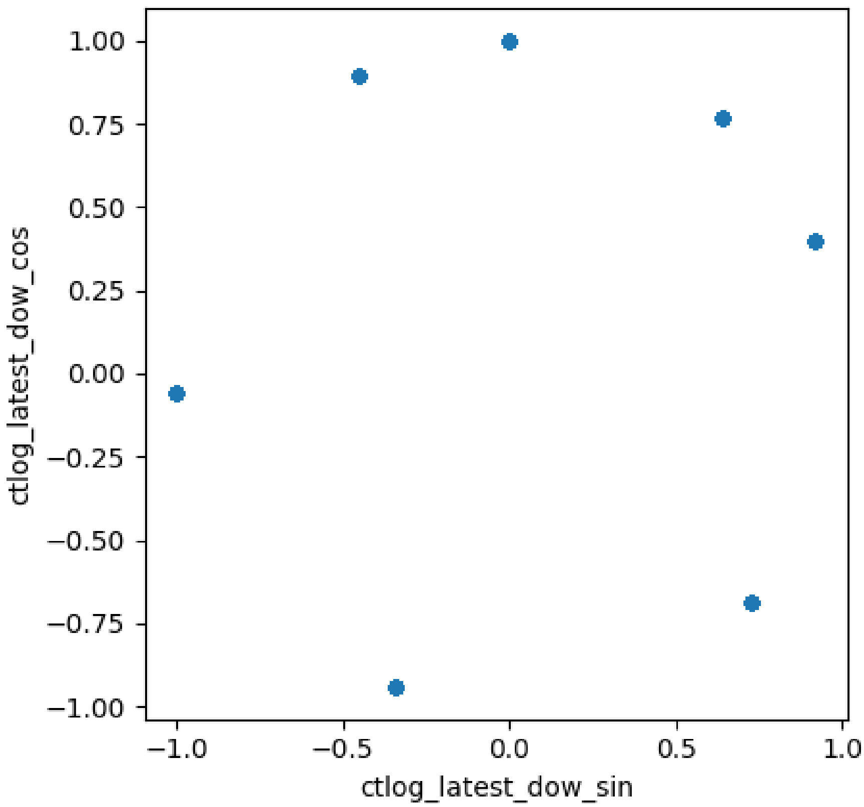
```
max                   13798.000000               0.918032
std                    3850.835626               0.659922

        whois_created_dow_cos  ctlog_earliest_dow_sin
ctlog_earliest_dow_cos
count            21549.000000            21549.000000
21549.000000  \
unique                    NaN                     NaN
NaN
top                       NaN                     NaN
NaN
freq                      NaN                     NaN
NaN
mean                 0.054288                0.095357
0.161451
min                 -0.940168               -0.998199                -
0.940168
25%                 -0.685567               -0.340712                -
0.685567
50%                  0.396506                0.000000
0.396506
75%                  0.767830                0.728010
0.892589
max                  1.000000                0.918032
1.000000
std                  0.736128                0.651782
0.734891

        ctlog_latest_dow_sin  ctlog_latest_dow_cos
count           21549.000000          21549.000000
unique                   NaN                   NaN
top                      NaN                   NaN
freq                     NaN                   NaN
mean                0.096253              0.255578
min                -0.998199             -0.940168
25%                -0.450871             -0.685567
50%                 0.000000              0.396506
75%                 0.728010              0.892589
max                 0.918032              1.000000
std                 0.651597              0.707728
```



whois_created
ctlog_earliest
whois_created_dayofweek
ctlog_earliest_dayofweek
domain_to_earliest_cert_delta
domain_to_latest_cert_delta


1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
ctlog_earliest_dow_cos
−1.0
−0.5
0.0
0.5
1.0
ctlog_earliest_dow_sin

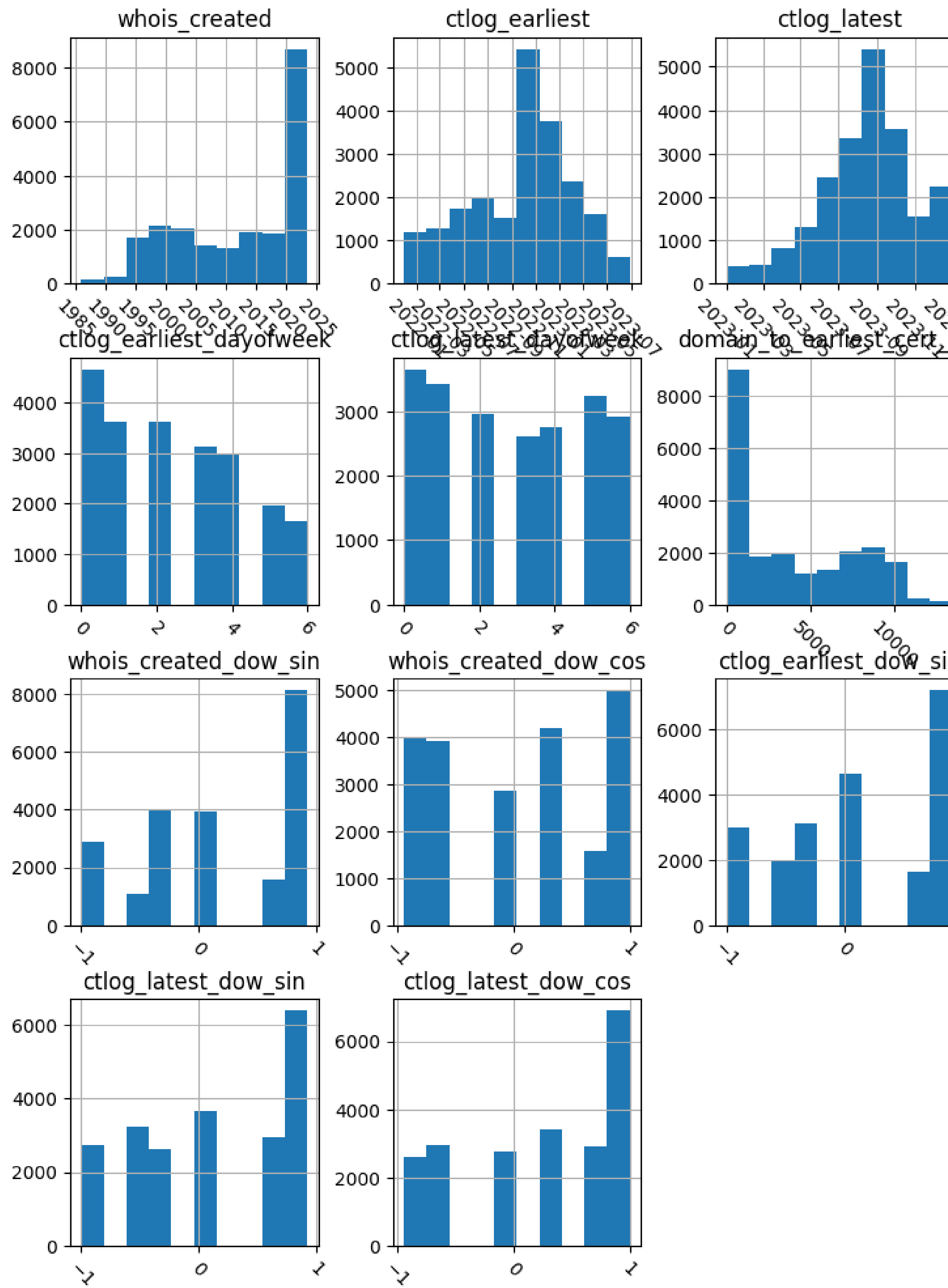

1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
whois_created_dow_cos
−1.0
−0.5
0.0
0.5
1.0
whois_created_dow_sin

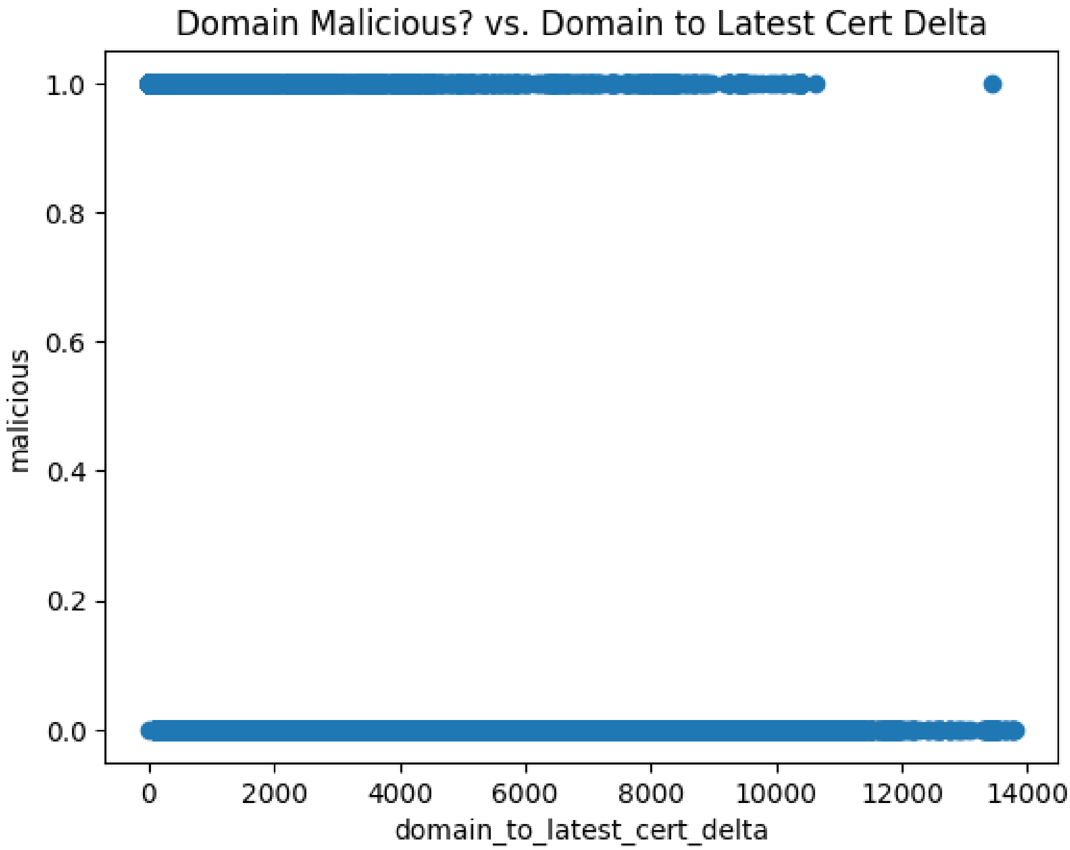


In [4]:

```
# Plot histograms
df.hist(figsize=(12,12), xrot=-45)

# Create a scatter plot of the data, showing malicious and benign domains
# plot the data as a scatter plot
plt.scatter(df["domain_to_earliest_cert_delta"], df["malicious"])
plt.xlabel("domain_to_earliest_cert_delta")
plt.ylabel("malicious")
plt.title("Domain Malicious? vs. Domain to Earliest Cert Delta")
plt.show()
# and for the latest
plt.scatter(df["domain_to_latest_cert_delta"], df["malicious"])
plt.xlabel("domain_to_latest_cert_delta")
plt.ylabel("malicious")
plt.title("Domain Malicious? vs. Domain to Latest Cert Delta")
plt.show()



click.echo(df.head())

# print a count of malicious and benign values
click.echo(df["malicious"].value_counts())
# deduplicate any rows, there shouldn't be any, but just in case
df = df.drop_duplicates()
click.echo(df["malicious"].value_counts())
```

```
click.echo(df.head())

X = df.drop(["malicious","domain", "ctlog_earliest", "ctlog_latest",
"whois_created", "whois_created_dayofweek", "ctlog_earliest_dayofweek"],
axis=1)
X = df.filter(combo_features)
y = df.filter(["malicious"])


click.echo(X.describe())
```


whois_created
ctlog_earliest
ctlog_latest
ctlog_earliest_dayofweek
ctlog_latest_dayofweek
domain_to_earliest_cert_
whois_created_dow_sin
whois_created_dow_cos
ctlog_earliest_dow_si
ctlog_latest_dow_sin
ctlog_latest_dow_cos
0
2000
4000
6000
8000
1000
3000
5000
1985
1990
1995
2000
2005
2010
2015
2020
2025
2
4
6
10000
-1
1




```
                                domain  malicious        whois_created  
0  i-db5p-cor001.api.p001.1drv.com      False 2013-08-05 18:33:50   \
4       soundcloud-pax.pandora.com      False 1993-12-28 05:00:00   
5         joolcomercializadora.com       True 2023-05-22 14:53:50   
6        createpdf-asr.acrobat.com      False 1999-03-16 05:00:00   
8                          popt.in      False 2016-05-14 16:58:55   

       ctlog_earliest         ctlog_latest  ctlog_wildcard  
0 2022-01-24 20:01:58 2023-06-08 20:46:06            True   \
4 2022-05-19 00:00:00 2023-06-19 23:59:59            True   
5 2022-04-06 22:23:24 2023-09-22 23:59:59           False   
6 2022-09-09 00:00:00 2023-10-10 23:59:59            True   
8 2023-01-07 20:36:15 2023-08-15 04:16:52           False   

   whois_created_dayofweek  ctlog_earliest_dayofweek  
ctlog_latest_dayofweek  
0                        0                          0  
3  \
4                        1                          3  
0  
5                        0                          2  
4  
6                        1                          4  
1  
8                        5                          5  
1  
```

```
   domain_to_earliest_cert_delta  domain_to_latest_cert_delta
0                         3095.0                       3595.0  \
4                        10369.0                      10766.0
5                          410.0                        124.0
6                         8578.0                       8975.0
8                         2430.0                       2649.0

   whois_created_dow_sin  whois_created_dow_cos  ctlog_earliest_dow_sin
0               0.000000               1.000000                0.000000  \
4               0.918032               0.396506               -0.340712
5               0.000000               1.000000                0.728010
6               0.918032               0.396506               -0.998199
8              -0.450871               0.892589               -0.450871

   ctlog_earliest_dow_cos  ctlog_latest_dow_sin  ctlog_latest_dow_cos
0                1.000000             -0.340712             -0.940168
4               -0.940168              0.000000              1.000000
5               -0.685567             -0.998199             -0.059997
6               -0.059997              0.918032              0.396506
8                0.892589              0.918032              0.396506
malicious
False    11739
True      9810
Name: count, dtype: int64
malicious
False    11739
True      9810
Name: count, dtype: int64
                            domain  malicious       whois_created
0  i-db5p-cor001.api.p001.1drv.com      False 2013-08-05 18:33:50  \
4       soundcloud-pax.pandora.com      False 1993-12-28 05:00:00
5         joolcomercializadora.com       True 2023-05-22 14:53:50
6        createpdf-asr.acrobat.com      False 1999-03-16 05:00:00
8                          popt.in      False 2016-05-14 16:58:55

       ctlog_earliest        ctlog_latest  ctlog_wildcard
0 2022-01-24 20:01:58 2023-06-08 20:46:06            True  \
4 2022-05-19 00:00:00 2023-06-19 23:59:59            True
5 2022-04-06 22:23:24 2023-09-22 23:59:59           False
6 2022-09-09 00:00:00 2023-10-10 23:59:59            True
8 2023-01-07 20:36:15 2023-08-15 04:16:52           False

   whois_created_dayofweek  ctlog_earliest_dayofweek
ctlog_latest_dayofweek
0                        0                         0
3  \
4                        1                         3
0
```

```
5                        0                          2
4
6                        1                          4
1
8                        5                          5
1

   domain_to_earliest_cert_delta  domain_to_latest_cert_delta
0                         3095.0                       3595.0  \
4                        10369.0                      10766.0
5                          410.0                        124.0
6                         8578.0                       8975.0
8                         2430.0                       2649.0

   whois_created_dow_sin  whois_created_dow_cos  ctlog_earliest_dow_sin
0               0.000000               1.000000                0.000000  \
4               0.918032               0.396506               -0.340712
5               0.000000               1.000000                0.728010
6               0.918032               0.396506               -0.998199
8              -0.450871               0.892589               -0.450871

   ctlog_earliest_dow_cos  ctlog_latest_dow_sin  ctlog_latest_dow_cos
0                1.000000             -0.340712             -0.940168
4               -0.940168              0.000000              1.000000
5               -0.685567             -0.998199             -0.059997
6               -0.059997              0.918032              0.396506
8                0.892589              0.918032              0.396506
       domain_to_earliest_cert_delta  whois_created_dow_sin
count                   21549.000000           21549.000000  \
mean                     3742.948397               0.140419
std                      3694.584062               0.659922
min                         0.000000              -0.998199
25%                       181.000000              -0.340712
50%                      2637.000000               0.000000
75%                      7078.000000               0.728010
max                     13445.000000               0.918032

       whois_created_dow_cos
count           21549.000000
mean                0.054288
std                 0.736128
min                -0.940168
25%                -0.685567
50%                 0.396506
75%                 0.767830
max                 1.000000
```

In [5]:

```
# convert y (malicious) to 1/0 int
y = y.astype('int')
```

```
# convert X["ctlog_wildcard"]  to 1/0 int
if "ctlog_wildcard" in X.columns:
    X["ctlog_wildcard"] = X["ctlog_wildcard"].astype('int')

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

X_train = sm.add_constant(X_train)
X_test = sm.add_constant(X_test)

smfit = sm.Logit(y_train,X_train).fit()

smfit.summary()
Optimization terminated successfully.
         Current function value: 0.372433
         Iterations 7
```

Out[5]:

Logit Regression Results

Dep. Variable:	malicious	**No. Observations:**	17239
Model:	Logit	**Df Residuals:**	17235
Method:	MLE	**Df Model:**	3
Date:	Tue, 08 Aug 2023	**Pseudo R-squ.:**	0.4598
Time:	19:12:39	**Log-Likelihood:**	-6420.4
converged:	True	**LL-Null:**	-11885.
Covariance Type:	nonrobust	**LLR p-value:**	0.000

	coef	**std err**	**z**	**P>|z|**	**[0.025**	**0.975]**
const	1.7800	0.031	57.529	0.000	1.719	1.841
domain_to_earliest_cert_delta	-0.0007	1.01e-05	-67.551	0.000	-0.001	-0.001
whois_created_dow_sin	0.1364	0.034	4.038	0.000	0.070	0.203
whois_created_dow_cos	-0.0606	0.030	-1.993	0.046	-0.120	-0.001

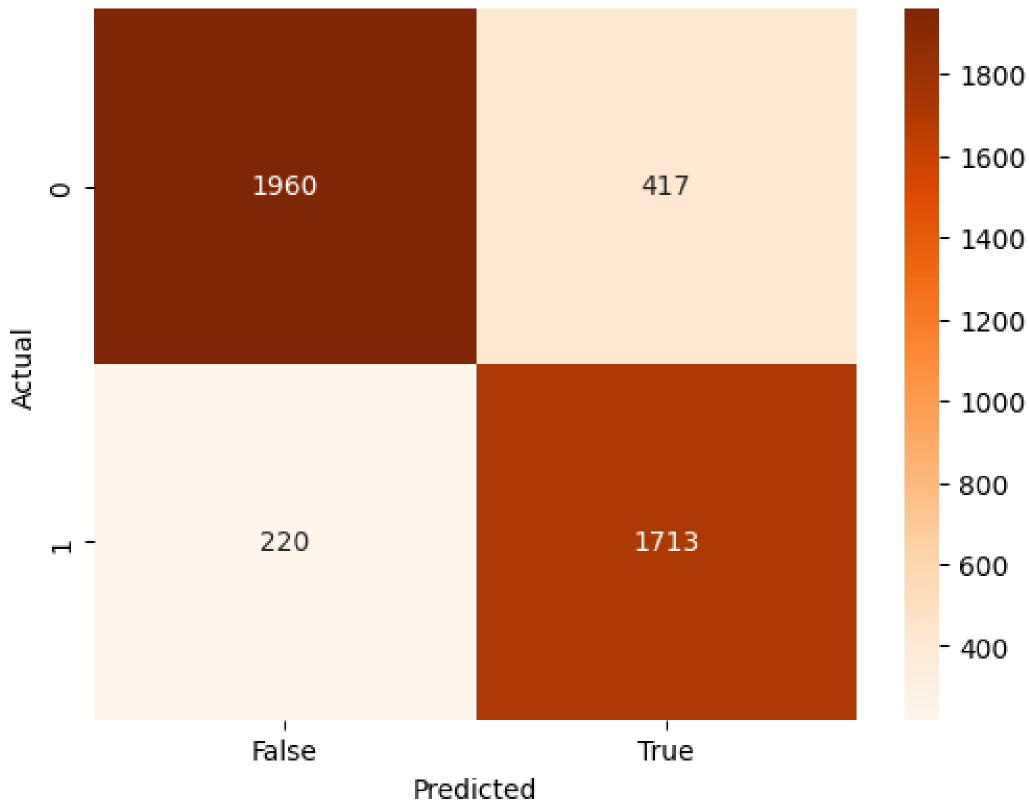
In [6]:

```
# Predict the malicious column using the test data
#add the incepts


y_predicted = smfit.predict(X_test)

# Present the results in a confusion matrix
confusion_matrix = confusion_matrix(y_test, y_predicted.round())
click.echo(confusion_matrix)

click.echo("Classification report:")
click.echo(classification_report(y_test, y_predicted.round()))

# Heatmap of confusion matrix
y_predicted

threshold = 0.5
```

```
y_predicted = [y > threshold for y in y_predicted]
data = {'Actual':    y_test.values.flatten(),
        'Predicted': y_predicted
        }

# Generate a confusion matrix and heatmap to evaluate the Type I and Type
II errors/ FP/FN etc.
df = pd.DataFrame(data, columns=['Actual','Predicted'])
confusion_matrix = pd.crosstab(df['Actual'], df['Predicted'],
rownames=['Actual'], colnames=['Predicted'])
fig = sns.heatmap(confusion_matrix, annot=True, cmap='Oranges', fmt='g')
fig
[[1960  417]
 [ 220 1713]]
Classification report:
              precision    recall  f1-score   support

           0       0.90      0.82      0.86      2377
           1       0.80      0.89      0.84      1933

    accuracy                           0.85      4310
   macro avg       0.85      0.86      0.85      4310
weighted avg       0.86      0.85      0.85      4310
```

Out[6]:

```
<Axes: xlabel='Predicted', ylabel='Actual'>
```



VIII. Feature Set G

In [1]:

```
import click
import pandas as pd
import glob
import os
#import sklearn
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report
from sklearn.preprocessing import StandardScaler
from scipy import stats
from sklearn.linear_model import LogisticRegression
import statsmodels.api as sm
import matplotlib.pyplot as plt
from datetime import datetime, date
# qqplot of the data
import seaborn as sns
import math
import numpy as np
import utils

combo_features = [
    'domain_to_earliest_cert_delta',
    'ctlog_earliest_dow_sin',
    'ctlog_earliest_dow_cos',
    'ctlog_wildcard',
    'whois_created_dow_sin',
    'whois_created_dow_cos']

path = "../data/"
filename = None

epoch = datetime(1970,1,1)
# open the training data file, from ../data/ for the most recent merged
training data file saved by prepare-training-data-parallel.py
full_path = path + "merged_20230705-104357_training_adorned-engineered.csv"
# get latest file matching the glob
if not filename:
    filename = max(glob.glob(full_path), key=os.path.getctime)
    click.echo(f"Using {filename} as training data")
    df = pd.read_csv(filename, sep=",")

# shuffle the rows
df = df.sample(frac=1, random_state=42).reset_index(drop=True)

click.echo(df.shape)
click.echo(df.columns)
```

```
""" # From prepare-training-data-parallel.py:
# Columns:
url
verification_time
domain
malicious
dateadded
whois_created
soa_timestamp
ctlog_earliest
ctlog_latest
ctlog_wildcard
whois_created_dayofweek,
ctlog_earliest_dayofweek,
domain_to_cert_delta
"""


# Data cleaning - there may be some epoch values in the whois_created
#  & ct_log_earliest columns, so we need to remove those rows

# convert the whois_created and ctlog_earliest, ctlog_latest columns to
datetime

df = df.loc[df["whois_created"] != ""]
df = df.loc[df["ctlog_earliest"] != ""]
df = df.loc[df["ctlog_latest"] != ""]

# some dates are have nanoseconds in them, so we need to remove the
nanosecond part
df["whois_created"] = df["whois_created"].str.split(".", n = 1, expand =
True)[0]
# some dates has additional timezone information, so we need to remove that
df["whois_created"] = df["whois_created"].apply(lambda x: " ".join(
str(x).split(" ")[:2]))


# convert the whois_created and ct_log_earliest columns to datetime
df = df.astype({"whois_created": 'datetime64[ns]', "ctlog_earliest":
'datetime64[ns]', "ctlog_latest": 'datetime64[ns]'})
df = df.loc[df["whois_created"].ne(pd.Timestamp('1970-01-01 00:00:00'))]
df = df.loc[df["ctlog_earliest"].ne(pd.Timestamp('1970-01-01 00:00:00'))]
df = df.loc[df["ctlog_latest"].ne(pd.Timestamp('1970-01-01 00:00:00'))]


# malicious is a bool
df['malicious'] = df['malicious'].astype('bool')
df['domain'] = df['domain'].astype('string')
```

```
Using ../data/merged_20230705-104357_training_adorned-engineered.csv as
training data
(35438, 13)
Index(['url', 'verification_time', 'domain', 'malicious', 'dateadded',
       'whois_created', 'soa_timestamp', 'ctlog_earliest', 'ctlog_latest',
       'ctlog_wildcard', 'whois_created_dayofweek',
'ctlog_earliest_dayofweek',
       'domain_to_cert_delta'],
      dtype='object')
```

In [2]:

```
#########
# Feature engineering
#########

# remove any rows where the whois_created or ctlog_earliest columns are
null
df = df.loc[df["whois_created"].notnull()]
df = df.loc[df["ctlog_earliest"].notnull()]

# if the data is missing the "whois_created_dayofweek" or
"ctlog_earliest_dayofweek" columns, then add them
# whois created day of the week 0 = Monday, 6 = Sunday
df["whois_created_dayofweek"] = df["whois_created"].apply(
        lambda x: x.weekday() if isinstance(x, date) else None
    )

# cert valid day of the week 0 = Monday, 6 = Sunday
df["ctlog_earliest_dayofweek"] = df["ctlog_earliest"].apply(
        lambda x: x.weekday() if isinstance(x, date) else None
    )

df["ctlog_latest_dayofweek"] = df["ctlog_latest"].apply(
        lambda x: x.weekday() if isinstance(x, date) else None
    )

# set data type of the day of week columns to int
df["whois_created_dayofweek"] =
df["whois_created_dayofweek"].astype("int64")
df["ctlog_earliest_dayofweek"] =
df["ctlog_earliest_dayofweek"].astype("int64")
df["ctlog_latest_dayofweek"] = df["ctlog_latest_dayofweek"].astype("int64")

# domain to cert delta
df["domain_to_earliest_cert_delta"] = df.apply(
        lambda x: utils.get_days_delta(
            x["ctlog_earliest"],
            x["whois_created"]
            if x["whois_created"] and x["ctlog_earliest"]
            else None,
```

```
        ),
        axis=1,
    )

# set the data type of the domain_to_cert_delta column to float
df["domain_to_earliest_cert_delta"] =
df["domain_to_earliest_cert_delta"].astype("float64")

# domain to latest cert delta
df["domain_to_latest_cert_delta"] = df.apply(
        lambda x: utils.get_days_delta(
            x["ctlog_latest"],
            x["whois_created"]
            if x["whois_created"] and x["ctlog_latest"]
            else None,
        ),
        axis=1,
    )

# set the data type of the domain_to_cert_delta column to float
df["domain_to_latest_cert_delta"] =
df["domain_to_latest_cert_delta"].astype("float64")

# drop columns we imported that we will never use
df = df.drop(columns=["url", "verification_time", "dateadded",
"soa_timestamp","domain_to_cert_delta"])

click.echo(df.head())
click.echo(df.describe(include='all'))
click.echo(df.dtypes)
                            domain  malicious       whois_created
0  i-db5p-cor001.api.p001.1drv.com      False 2013-08-05 18:33:50  \
4       soundcloud-pax.pandora.com      False 1993-12-28 05:00:00
5         joolcomercializadora.com       True 2023-05-22 14:53:50
6        createpdf-asr.acrobat.com      False 1999-03-16 05:00:00
8                          popt.in      False 2016-05-14 16:58:55

       ctlog_earliest        ctlog_latest  ctlog_wildcard
0 2022-01-24 20:01:58 2023-06-08 20:46:06            True  \
4 2022-05-19 00:00:00 2023-06-19 23:59:59            True
5 2022-04-06 22:23:24 2023-09-22 23:59:59           False
6 2022-09-09 00:00:00 2023-10-10 23:59:59            True
8 2023-01-07 20:36:15 2023-08-15 04:16:52           False

   whois_created_dayofweek  ctlog_earliest_dayofweek
ctlog_latest_dayofweek
0                        0                          0
3  \
```

```
4                       1                         3
0
5                       0                         2
4
6                       1                         4
1
8                       5                         5
1

   domain_to_earliest_cert_delta  domain_to_latest_cert_delta
0                        -3095.0                      -3595.0
4                       -10369.0                     -10766.0
5                          410.0                       -124.0
6                        -8578.0                      -8975.0
8                        -2430.0                      -2649.0
                  domain malicious                  whois_created
count              21549     21549                          21549  \
unique             21536         2                            NaN
top     www.mediafire.com     False                            NaN
freq                   2     11739                            NaN
mean                 NaN       NaN  2012-10-03 12:56:32.335050496
min                  NaN       NaN            1986-01-09 00:00:00
25%                  NaN       NaN            2003-05-25 13:35:05
50%                  NaN       NaN            2015-05-07 23:56:05
75%                  NaN       NaN            2023-03-20 15:03:16
max                  NaN       NaN            2023-07-03 08:21:24
std                  NaN       NaN                            NaN

                        ctlog_earliest                   ctlog_latest
count                            21549                          21549  \
unique                             NaN                            NaN
top                                NaN                            NaN
freq                               NaN                            NaN
mean    2022-09-26 15:45:50.943570432  2023-08-14 17:49:06.400900352
min               2021-11-30 05:24:28            2023-01-01 18:42:11
25%               2022-06-24 13:47:12            2023-07-02 08:11:07
50%               2022-10-18 21:00:14            2023-08-21 21:40:11
75%               2022-12-14 00:00:00            2023-09-21 19:41:38
max               2023-06-28 04:36:22            2023-12-31 23:59:59
std                                NaN                            NaN

       ctlog_wildcard  whois_created_dayofweek  ctlog_earliest_dayofweek
count           21549             21549.000000              21549.000000  \
unique              2                      NaN                       NaN
top             False                      NaN                       NaN
freq            13032                      NaN                       NaN
mean              NaN                 2.332823                  2.399462
min               NaN                 0.000000                  0.000000
25%               NaN                 1.000000                  1.000000
```

```
50%                NaN                 2.000000                  2.000000
75%                NaN                 4.000000                  4.000000
max                NaN                 6.000000                  6.000000
std                NaN                 1.775043                  1.897252

        ctlog_latest_dayofweek  domain_to_earliest_cert_delta
count             21549.000000                   21549.000000  \
unique                     NaN                            NaN
top                        NaN                            NaN
freq                       NaN                            NaN
mean                  2.873080                   -3645.602070
min                   0.000000                  -13445.000000
25%                   1.000000                   -7078.000000
50%                   3.000000                   -2637.000000
75%                   5.000000                      69.000000
max                   6.000000                     524.000000
std                   2.057394                    3790.677119

        domain_to_latest_cert_delta
count                  21549.000000
unique                          NaN
top                             NaN
freq                            NaN
mean                   -3967.678222
min                   -13798.000000
25%                    -7421.000000
50%                    -3009.000000
75%                     -144.000000
max                      135.000000
std                     3852.703681
domain                            string[python]
malicious                                   bool
whois_created                     datetime64[ns]
ctlog_earliest                    datetime64[ns]
ctlog_latest                      datetime64[ns]
ctlog_wildcard                              bool
whois_created_dayofweek                    int64
ctlog_earliest_dayofweek                   int64
ctlog_latest_dayofweek                     int64
domain_to_earliest_cert_delta            float64
domain_to_latest_cert_delta              float64
dtype: object
```

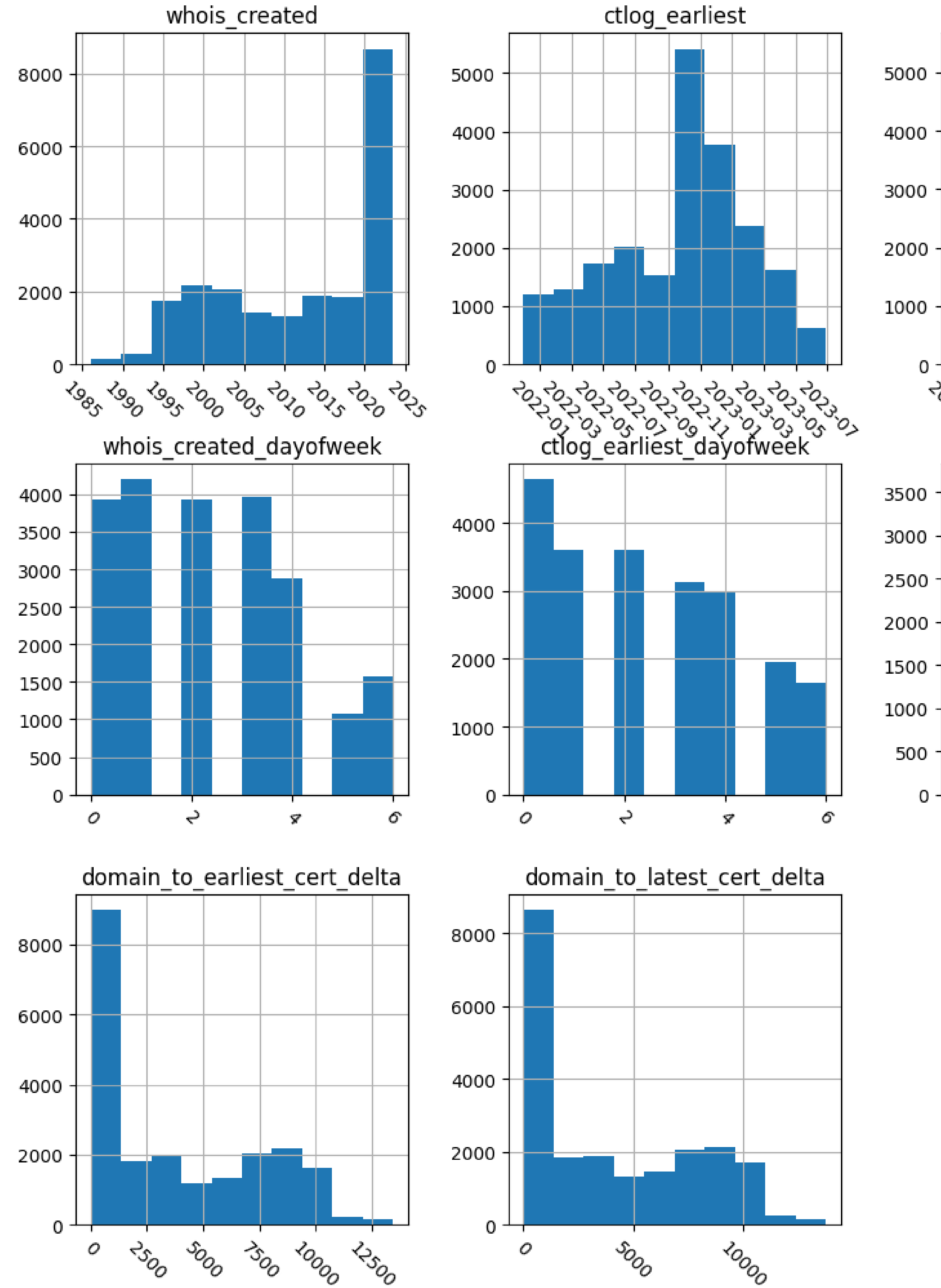
In [3]:

```
# absolute value of the domain_to_cert_delta
df["domain_to_earliest_cert_delta"] =
abs(df["domain_to_earliest_cert_delta"])
df["domain_to_latest_cert_delta"] = abs(df["domain_to_latest_cert_delta"])

df.hist(figsize=(12,12), xrot=-45)
```

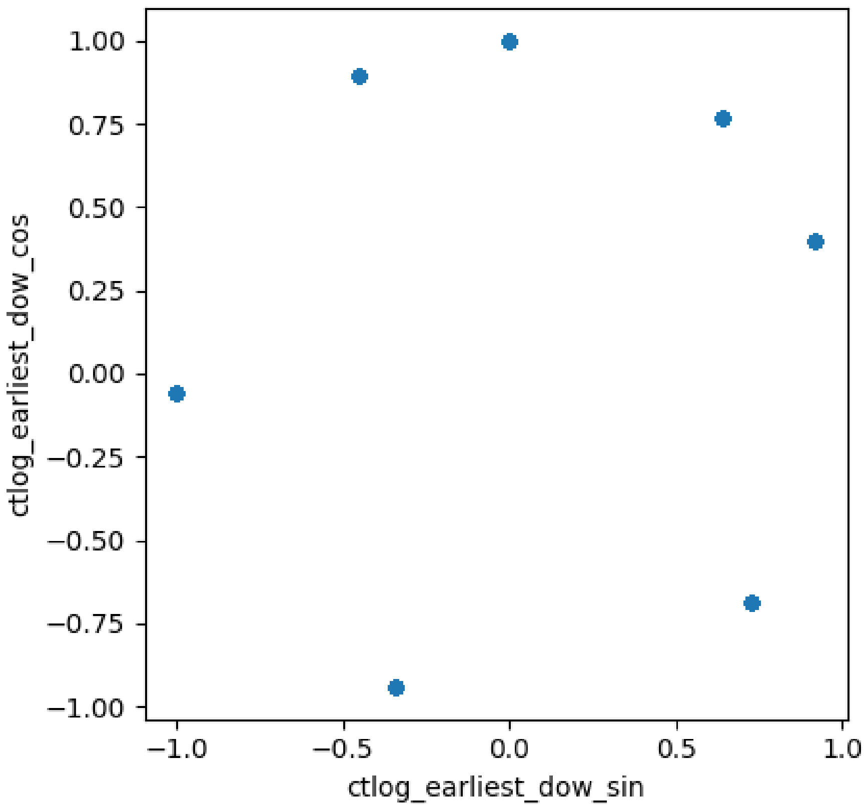
```
col_index = df.columns.get_loc("whois_created_dayofweek")
df["whois_created_dow_sin"] = df.apply(lambda row:
math.sin(360/7*(row[col_index])),axis=1)
df["whois_created_dow_cos"] = df.apply(lambda row:
math.cos(360/7*(row[col_index])),axis=1)

col_index = df.columns.get_loc("ctlog_earliest_dayofweek")
df["ctlog_earliest_dow_sin"] = df.apply(lambda row:
math.sin(360/7*(row[col_index])),axis=1)
df["ctlog_earliest_dow_cos"] = df.apply(lambda row:
math.cos(360/7*(row[col_index])),axis=1)

col_index = df.columns.get_loc("ctlog_latest_dayofweek")
df["ctlog_latest_dow_sin"] = df.apply(lambda row:
math.sin(360/7*(row[col_index])),axis=1)
df["ctlog_latest_dow_cos"] = df.apply(lambda row:
math.cos(360/7*(row[col_index])),axis=1)


df.plot.scatter(x="ctlog_earliest_dow_sin",
y="ctlog_earliest_dow_cos").set_aspect('equal')
df.plot.scatter(x="whois_created_dow_sin",
y="whois_created_dow_cos").set_aspect('equal')
df.plot.scatter(x="ctlog_latest_dow_sin",
y="ctlog_latest_dow_cos").set_aspect('equal')

"""
# add one hot encode ctlog_earliest_not_before_dayofweek
df = pd.get_dummies(
        df,
        columns=["ctlog_earliest_dayofweek"],
        prefix=["ctlog_earliest_dow"],
    )
# add one hot encode whois_created_dayofweek to columns
df = pd.get_dummies(
        df, columns=["whois_created_dayofweek"], prefix="whois_dow"
    )
    """

# Summary statistics
click.echo(df.describe(include='all'))
                   domain malicious                     whois_created
count               21549     21549                             21549  \
unique              21536         2                               NaN
top     www.mediafire.com     False                               NaN
freq                    2     11739                               NaN
mean                  NaN       NaN  2012-10-03 12:56:32.335050496
min                   NaN       NaN            1986-01-09 00:00:00
```

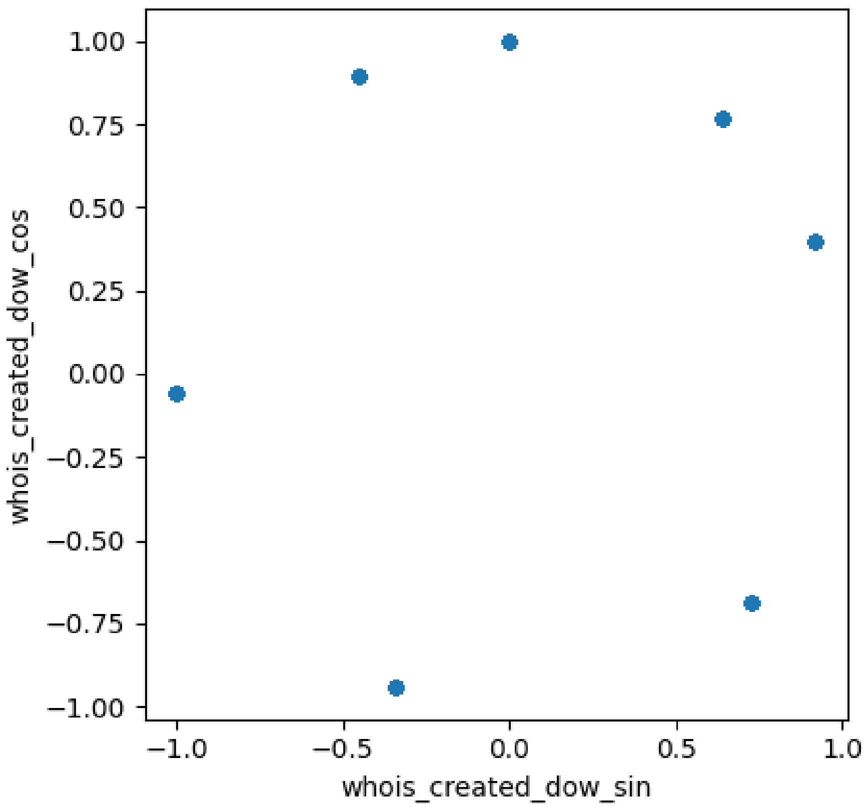
```
25%                   NaN        NaN            2003-05-25 13:35:05
50%                   NaN        NaN            2015-05-07 23:56:05
75%                   NaN        NaN            2023-03-20 15:03:16
max                   NaN        NaN            2023-07-03 08:21:24
std                   NaN        NaN                            NaN

                     ctlog_earliest                   ctlog_latest
count                         21549                          21549  \
unique                          NaN                            NaN
top                             NaN                            NaN
freq                            NaN                            NaN
mean    2022-09-26 15:45:50.943570432  2023-08-14 17:49:06.400900352
min               2021-11-30 05:24:28            2023-01-01 18:42:11
25%               2022-06-24 13:47:12            2023-07-02 08:11:07
50%               2022-10-18 21:00:14            2023-08-21 21:40:11
75%               2022-12-14 00:00:00            2023-09-21 19:41:38
max               2023-06-28 04:36:22            2023-12-31 23:59:59
std                             NaN                            NaN

       ctlog_wildcard  whois_created_dayofweek  ctlog_earliest_dayofweek
count           21549             21549.000000              21549.000000  \
unique              2                      NaN                       NaN
top             False                      NaN                       NaN
freq            13032                      NaN                       NaN
mean              NaN                 2.332823                  2.399462
min               NaN                 0.000000                  0.000000
25%               NaN                 1.000000                  1.000000
50%               NaN                 2.000000                  2.000000
75%               NaN                 4.000000                  4.000000
max               NaN                 6.000000                  6.000000
std               NaN                 1.775043                  1.897252

        ctlog_latest_dayofweek  domain_to_earliest_cert_delta
count             21549.000000                   21549.000000  \
unique                     NaN                            NaN
top                        NaN                            NaN
freq                       NaN                            NaN
mean                  2.873080                    3742.948397
min                   0.000000                       0.000000
25%                   1.000000                     181.000000
50%                   3.000000                    2637.000000
75%                   5.000000                    7078.000000
max                   6.000000                   13445.000000
std                   2.057394                    3694.584062

        domain_to_latest_cert_delta  whois_created_dow_sin
count                  21549.000000           21549.000000  \
unique                          NaN                    NaN
top                             NaN                    NaN
```

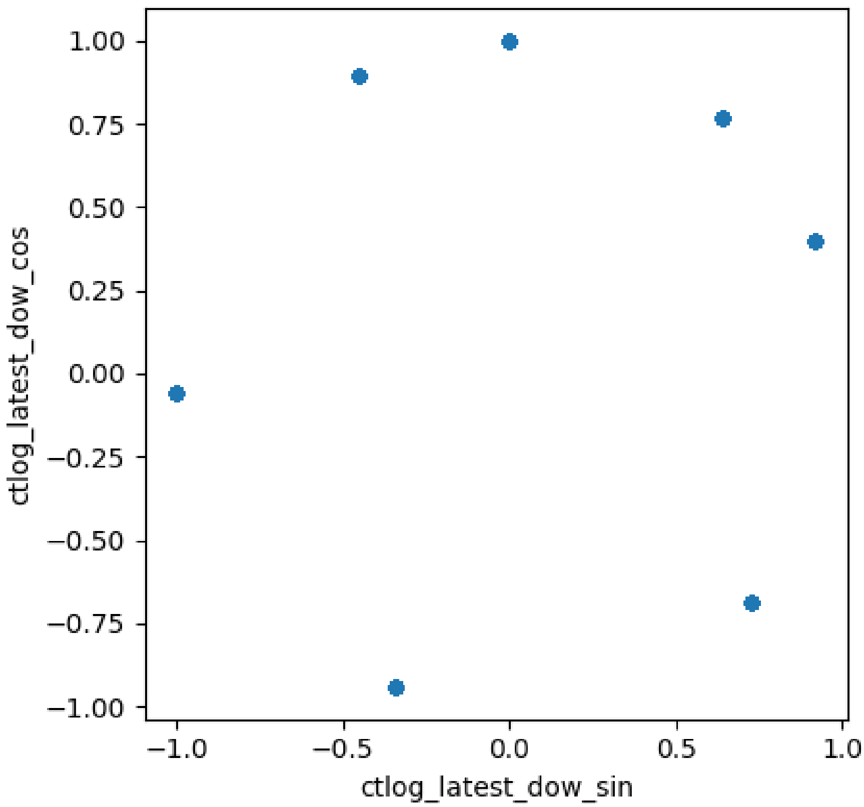
```
freq                           NaN                     NaN
mean                   3969.491206                0.140419
min                       0.000000               -0.998199
25%                     144.000000               -0.340712
50%                    3009.000000                0.000000
75%                    7421.000000                0.728010
max                   13798.000000                0.918032
std                    3850.835626                0.659922

        whois_created_dow_cos  ctlog_earliest_dow_sin
ctlog_earliest_dow_cos
count            21549.000000            21549.000000
21549.000000  \
unique                    NaN                     NaN
NaN
top                       NaN                     NaN
NaN
freq                      NaN                     NaN
NaN
mean                 0.054288                0.095357
0.161451
min                 -0.940168               -0.998199               -
0.940168
25%                 -0.685567               -0.340712               -
0.685567
50%                  0.396506                0.000000
0.396506
75%                  0.767830                0.728010
0.892589
max                  1.000000                0.918032
1.000000
std                  0.736128                0.651782
0.734891

        ctlog_latest_dow_sin  ctlog_latest_dow_cos
count           21549.000000          21549.000000
unique                   NaN                   NaN
top                      NaN                   NaN
freq                     NaN                   NaN
mean                0.096253              0.255578
min                -0.998199             -0.940168
25%                -0.450871             -0.685567
50%                 0.000000              0.396506
75%                 0.728010              0.892589
max                 0.918032              1.000000
std                 0.651597              0.707728
```



whois_created
0
2000
4000
6000
8000
1985
1990
1995
2000
2005
2010
2015
2020
2025
ctlog_earliest
0
1000
2000
3000
4000
5000
2022-01
2022-03
2022-05
2022-07
2022-09
2022-11
2023-01
2023-03
2023-05
2023-07
whois_created_dayofweek
0
500
1000
1500
2000
2500
3000
3500
4000
0
2
4
6
ctlog_earliest_dayofweek
0
1000
2000
3000
4000
0
2
4
6
domain_to_earliest_cert_delta
0
2000
4000
6000
8000
0
2500
5000
7500
10000
12500
domain_to_latest_cert_delta
0
2000
4000
6000
8000
0
5000
10000


1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
ctlog_earliest_dow_cos
−1.0
−0.5
0.0
0.5
1.0
ctlog_earliest_dow_sin

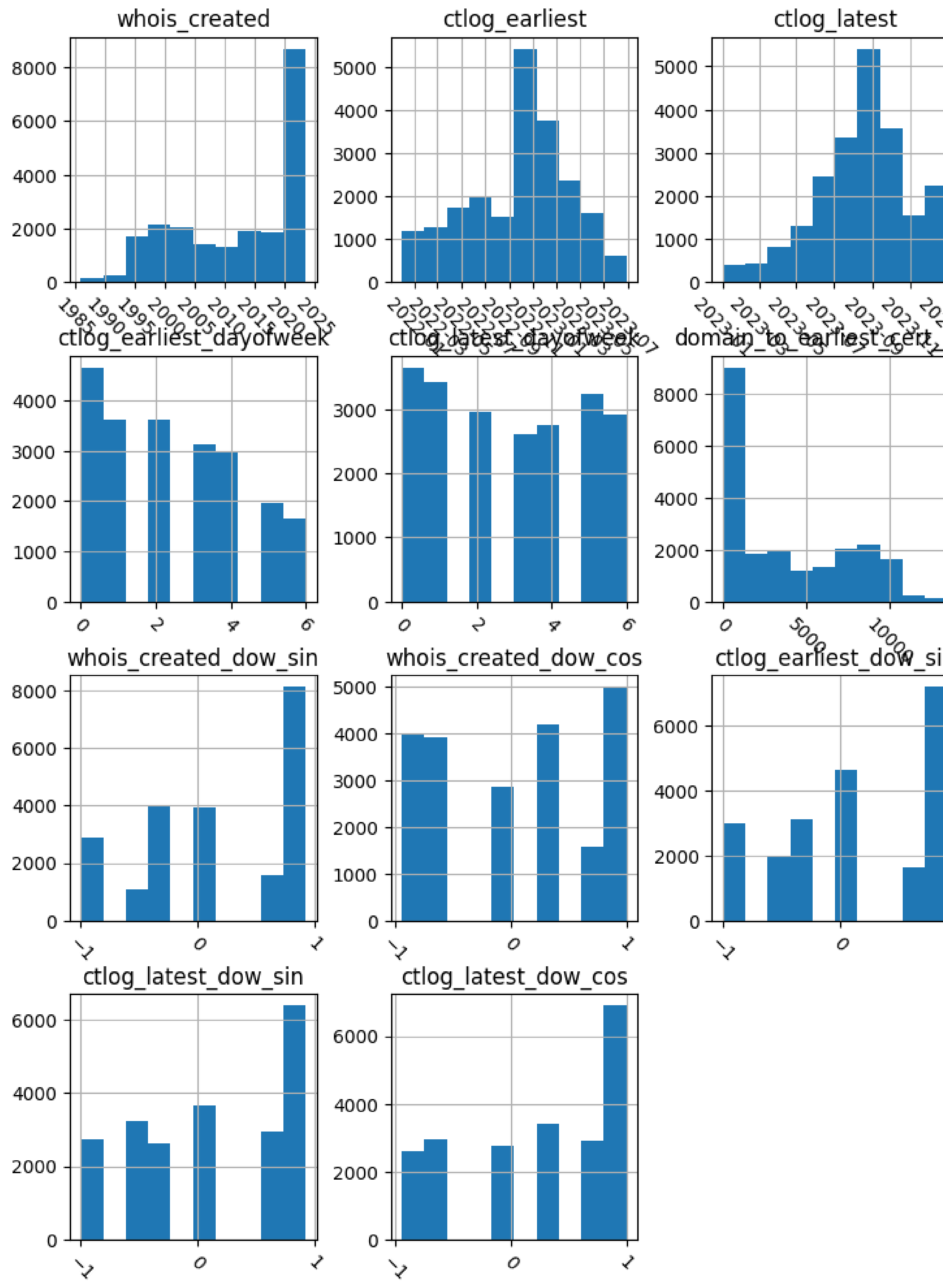

1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
whois_created_dow_cos
−1.0
−0.5
0.0
0.5
1.0
whois_created_dow_sin

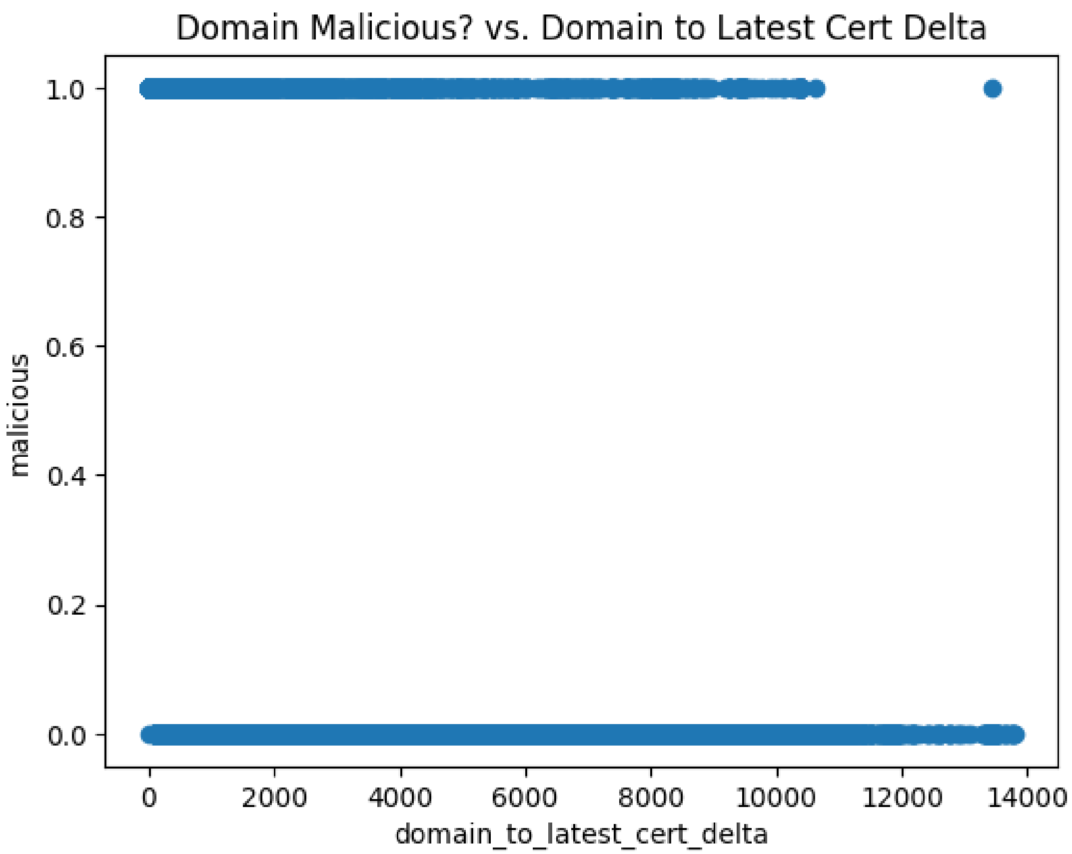


In [4]:

```
# Plot histograms
df.hist(figsize=(12,12), xrot=-45)

# Create a scatter plot of the data, showing malicious and benign domains
# plot the data as a scatter plot
plt.scatter(df["domain_to_earliest_cert_delta"], df["malicious"])
plt.xlabel("domain_to_earliest_cert_delta")
plt.ylabel("malicious")
plt.title("Domain Malicious? vs. Domain to Earliest Cert Delta")
plt.show()
# and for the latest
plt.scatter(df["domain_to_latest_cert_delta"], df["malicious"])
plt.xlabel("domain_to_latest_cert_delta")
plt.ylabel("malicious")
plt.title("Domain Malicious? vs. Domain to Latest Cert Delta")
plt.show()



click.echo(df.head())

# print a count of malicious and benign values
click.echo(df["malicious"].value_counts())
# deduplicate any rows, there shouldn't be any, but just in case
df = df.drop_duplicates()
click.echo(df["malicious"].value_counts())
```

```
click.echo(df.head())

X = df.drop(["malicious","domain", "ctlog_earliest", "ctlog_latest",
"whois_created", "whois_created_dayofweek", "ctlog_earliest_dayofweek"],
axis=1)
X = X.filter(combo_features)
y = df.filter(["malicious"])


click.echo(X.describe())
```


whois_created
ctlog_earliest
ctlog_latest
ctlog_earliest_dayofweek
ctlog_latest_dayofweek
domain_to_earliest_cert_
whois_created_dow_sin
whois_created_dow_cos
ctlog_earliest_dow_si
ctlog_latest_dow_sin
ctlog_latest_dow_cos
1985
1990
1995
2000
2005
2010
2015
2020
2025
0
2
4
6
5000
10000
-1
0
1




```
                                domain  malicious        whois_created
0  i-db5p-cor001.api.p001.1drv.com      False 2013-08-05 18:33:50   \
4        soundcloud-pax.pandora.com      False 1993-12-28 05:00:00
5          joolcomercializadora.com       True 2023-05-22 14:53:50
6         createpdf-asr.acrobat.com      False 1999-03-16 05:00:00
8                           popt.in      False 2016-05-14 16:58:55

       ctlog_earliest         ctlog_latest  ctlog_wildcard  \
0 2022-01-24 20:01:58 2023-06-08 20:46:06            True
4 2022-05-19 00:00:00 2023-06-19 23:59:59            True
5 2022-04-06 22:23:24 2023-09-22 23:59:59           False
6 2022-09-09 00:00:00 2023-10-10 23:59:59            True
8 2023-01-07 20:36:15 2023-08-15 04:16:52           False

   whois_created_dayofweek  ctlog_earliest_dayofweek
ctlog_latest_dayofweek
0                        0                          0
3  \
4                        1                          3
0
5                        0                          2
4
6                        1                          4
1
8                        5                          5
1
```

```
   domain_to_earliest_cert_delta  domain_to_latest_cert_delta
0                         3095.0                       3595.0  \
4                        10369.0                      10766.0
5                          410.0                        124.0
6                         8578.0                       8975.0
8                         2430.0                       2649.0

   whois_created_dow_sin  whois_created_dow_cos  ctlog_earliest_dow_sin
0               0.000000               1.000000                0.000000  \
4               0.918032               0.396506               -0.340712
5               0.000000               1.000000                0.728010
6               0.918032               0.396506               -0.998199
8              -0.450871               0.892589               -0.450871

   ctlog_earliest_dow_cos  ctlog_latest_dow_sin  ctlog_latest_dow_cos
0                1.000000             -0.340712             -0.940168
4               -0.940168              0.000000              1.000000
5               -0.685567             -0.998199             -0.059997
6               -0.059997              0.918032              0.396506
8                0.892589              0.918032              0.396506
malicious
False    11739
True      9810
Name: count, dtype: int64
malicious
False    11739
True      9810
Name: count, dtype: int64
                            domain  malicious       whois_created
0  i-db5p-cor001.api.p001.1drv.com      False 2013-08-05 18:33:50  \
4       soundcloud-pax.pandora.com      False 1993-12-28 05:00:00
5         joolcomercializadora.com       True 2023-05-22 14:53:50
6        createpdf-asr.acrobat.com      False 1999-03-16 05:00:00
8                          popt.in      False 2016-05-14 16:58:55

       ctlog_earliest        ctlog_latest  ctlog_wildcard
0 2022-01-24 20:01:58 2023-06-08 20:46:06            True  \
4 2022-05-19 00:00:00 2023-06-19 23:59:59            True
5 2022-04-06 22:23:24 2023-09-22 23:59:59           False
6 2022-09-09 00:00:00 2023-10-10 23:59:59            True
8 2023-01-07 20:36:15 2023-08-15 04:16:52           False

   whois_created_dayofweek  ctlog_earliest_dayofweek
ctlog_latest_dayofweek
0                        0                         0
3  \
4                        1                         3
0
```

```
5                    0                          2
4
6                    1                          4
1
8                    5                          5
1

   domain_to_earliest_cert_delta  domain_to_latest_cert_delta
0                         3095.0                       3595.0  \
4                        10369.0                      10766.0
5                          410.0                        124.0
6                         8578.0                       8975.0
8                         2430.0                       2649.0

   whois_created_dow_sin  whois_created_dow_cos  ctlog_earliest_dow_sin
0               0.000000               1.000000                0.000000  \
4               0.918032               0.396506               -0.340712
5               0.000000               1.000000                0.728010
6               0.918032               0.396506               -0.998199
8              -0.450871               0.892589               -0.450871

   ctlog_earliest_dow_cos  ctlog_latest_dow_sin  ctlog_latest_dow_cos
0                1.000000             -0.340712             -0.940168
4               -0.940168              0.000000              1.000000
5               -0.685567             -0.998199             -0.059997
6               -0.059997              0.918032              0.396506
8                0.892589              0.918032              0.396506
       domain_to_earliest_cert_delta  ctlog_earliest_dow_sin
count                   21549.000000            21549.000000  \
mean                     3742.948397                0.095357
std                      3694.584062                0.651782
min                         0.000000               -0.998199
25%                       181.000000               -0.340712
50%                      2637.000000                0.000000
75%                      7078.000000                0.728010
max                     13445.000000                0.918032

       ctlog_earliest_dow_cos  whois_created_dow_sin  whois_created_dow_cos
count            21549.000000           21549.000000           21549.000000
mean                 0.161451               0.140419               0.054288
std                  0.734891               0.659922               0.736128
min                 -0.940168              -0.998199              -0.940168
25%                 -0.685567              -0.340712              -0.685567
50%                  0.396506               0.000000               0.396506
75%                  0.892589               0.728010               0.767830
max                  1.000000               0.918032               1.000000
```

In [5]:

```
# convert y (malicious) to 1/0 int
y = y.astype('int')
```

```
# convert X["ctlog_wildcard"]  to 1/0 int
if "ctlog_wildcard" in X.columns:
    X["ctlog_wildcard"] = X["ctlog_wildcard"].astype('int')

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

X_train = sm.add_constant(X_train)
X_test = sm.add_constant(X_test)

smfit = sm.Logit(y_train,X_train).fit()

smfit.summary()
Optimization terminated successfully.
         Current function value: 0.354173
         Iterations 7
```

Out[5]:

Logit Regression Results

Dep. Variable:	malicious	**No. Observations:**	17239
Model:	Logit	**Df Residuals:**	17232
Method:	MLE	**Df Model:**	6
Date:	Tue, 08 Aug 2023	**Pseudo R-squ.:**	0.4863
Time:	19:11:43	**Log-Likelihood:**	-6105.6
converged:	True	**LL-Null:**	-11885.
Covariance Type:	nonrobust	**LLR p-value:**	0.000

	coef	**std err**	**z**	**P>|z|**	**[0.025**	**0.975]**
const	2.0846	0.036	58.141	0.000	2.014	2.155
domain_to_earliest_cert_delta	-0.0006	9.93e-06	-61.246	0.000	-0.001	-0.001
ctlog_earliest_dow_sin	0.1365	0.035	3.883	0.000	0.068	0.205
ctlog_earliest_dow_cos	-0.1844	0.032	-5.832	0.000	-0.246	-0.122
ctlog_wildcard	-1.1991	0.049	-24.579	0.000	-1.295	-1.104
whois_created_dow_sin	0.1103	0.035	3.181	0.001	0.042	0.178
whois_created_dow_cos	-0.0430	0.032	-1.362	0.173	-0.105	0.019

In [6]:

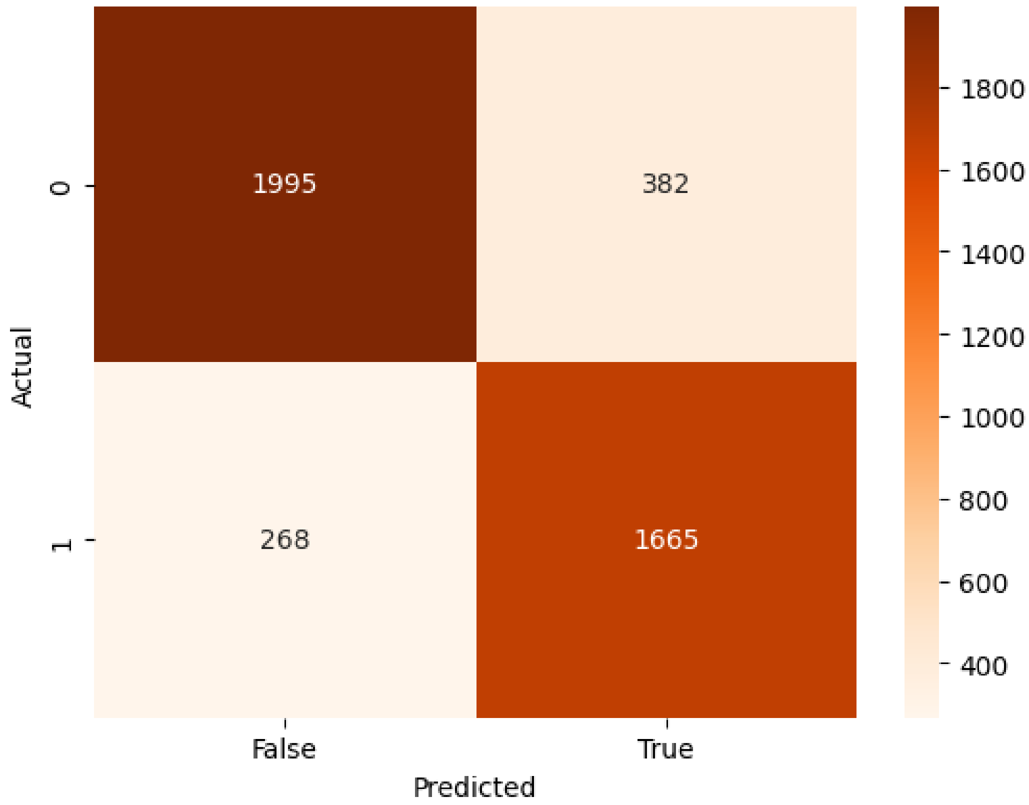
```
# Predict the malicious column using the test data
#add the incepts


y_predicted = smfit.predict(X_test)

# Present the results in a confusion matrix
confusion_matrix = confusion_matrix(y_test, y_predicted.round())
click.echo(confusion_matrix)

click.echo("Classification report:")
click.echo(classification_report(y_test, y_predicted.round()))

# Heatmap of confusion matrix
```

```
y_predicted

threshold = 0.5
y_predicted = [y > threshold for y in y_predicted]
data = {'Actual':    y_test.values.flatten(),
        'Predicted': y_predicted
        }

# Generate a confusion matrix and heatmap to evaluate the Type I and Type
II errors/ FP/FN etc.
df = pd.DataFrame(data, columns=['Actual','Predicted'])
confusion_matrix = pd.crosstab(df['Actual'], df['Predicted'],
rownames=['Actual'], colnames=['Predicted'])
fig = sns.heatmap(confusion_matrix, annot=True, cmap='Oranges', fmt='g')
fig
```

```
[[1995  382]
 [ 268 1665]]
Classification report:
              precision    recall  f1-score   support

           0       0.88      0.84      0.86      2377
           1       0.81      0.86      0.84      1933

    accuracy                           0.85      4310
   macro avg       0.85      0.85      0.85      4310
weighted avg       0.85      0.85      0.85      4310
```

Out[6]:

```
<Axes: xlabel='Predicted', ylabel='Actual'>
```


1995
382
268
1665
0
1
Actual
False
True
Predicted
1800
1600
1400
1200
1000
800
600
400

IX. Feature Set H

In [1]:

```
import click
import pandas as pd
import glob
import os
#import sklearn
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report
from sklearn.preprocessing import StandardScaler
from scipy import stats
from sklearn.linear_model import LogisticRegression
import statsmodels.api as sm
import matplotlib.pyplot as plt
from datetime import datetime, date
# qqplot of the data
import seaborn as sns
import math
import numpy as np
import utils


combo_features =[
    'domain_to_earliest_cert_delta',
    'ctlog_earliest_dow_sin',
    'ctlog_earliest_dow_cos',
    'ctlog_wildcard',
    'whois_created_dow_sin',
    'whois_created_dow_cos',
    'domain_to_latest_cert_delta',
    'ctlog_latest_dow_sin',
    'ctlog_latest_dow_cos'
    ]

path = "../data/"
filename = None

epoch = datetime(1970,1,1)
# open the training data file, from ../data/ for the most recent merged
training data file saved by prepare-training-data-parallel.py
full_path = path + "merged_20230705-104357_training_adorned-engineered.csv"
# get latest file matching the glob
if not filename:
    filename = max(glob.glob(full_path), key=os.path.getctime)
    click.echo(f"Using {filename} as training data")
    df = pd.read_csv(filename, sep=",")

# shuffle the rows
```

```
df = df.sample(frac=1, random_state=42).reset_index(drop=True)

click.echo(df.shape)
click.echo(df.columns)

""" # From prepare-training-data-parallel.py:
# Columns:
url
verification_time
domain
malicious
dateadded
whois_created
soa_timestamp
ctlog_earliest
ctlog_latest
ctlog_wildcard
whois_created_dayofweek,
ctlog_earliest_dayofweek,
domain_to_cert_delta
"""


# Data cleaning - there may be some epoch values in the whois_created
#  & ct_log_earliest columns, so we need to remove those rows

# convert the whois_created and ctlog_earliest, ctlog_latest columns to
datetime

df = df.loc[df["whois_created"] != ""]
df = df.loc[df["ctlog_earliest"] != ""]
df = df.loc[df["ctlog_latest"] != ""]

# some dates are have nanoseconds in them, so we need to remove the
nanosecond part
df["whois_created"] = df["whois_created"].str.split(".", n = 1, expand =
True)[0]
# some dates has additional timezone information, so we need to remove that
df["whois_created"] = df["whois_created"].apply(lambda x: " ".join(
str(x).split(" ")[:2]))


# convert the whois_created and ct_log_earliest columns to datetime
df = df.astype({"whois_created": 'datetime64[ns]', "ctlog_earliest":
'datetime64[ns]', "ctlog_latest": 'datetime64[ns]'})
df = df.loc[df["whois_created"].ne(pd.Timestamp('1970-01-01 00:00:00'))]
df = df.loc[df["ctlog_earliest"].ne(pd.Timestamp('1970-01-01 00:00:00'))]
df = df.loc[df["ctlog_latest"].ne(pd.Timestamp('1970-01-01 00:00:00'))]
```

```
# malicious is a bool
df['malicious'] = df['malicious'].astype('bool')
df['domain'] = df['domain'].astype('string')
Using ../data/merged_20230705-104357_training_adorned-engineered.csv as
training data
(35438, 13)
Index(['url', 'verification_time', 'domain', 'malicious', 'dateadded',
       'whois_created', 'soa_timestamp', 'ctlog_earliest', 'ctlog_latest',
       'ctlog_wildcard', 'whois_created_dayofweek',
'ctlog_earliest_dayofweek',
       'domain_to_cert_delta'],
      dtype='object')
```

In [2]:

```
#########
# Feature engineering
#########

# remove any rows where the whois_created or ctlog_earliest columns are
null
df = df.loc[df["whois_created"].notnull()]
df = df.loc[df["ctlog_earliest"].notnull()]

# if the data is missing the "whois_created_dayofweek" or
"ctlog_earliest_dayofweek" columns, then add them
# whois created day of the week 0 = Monday, 6 = Sunday
df["whois_created_dayofweek"] = df["whois_created"].apply(
        lambda x: x.weekday() if isinstance(x, date) else None
    )

# cert valid day of the week 0 = Monday, 6 = Sunday
df["ctlog_earliest_dayofweek"] = df["ctlog_earliest"].apply(
        lambda x: x.weekday() if isinstance(x, date) else None
    )

df["ctlog_latest_dayofweek"] = df["ctlog_latest"].apply(
        lambda x: x.weekday() if isinstance(x, date) else None
    )

# set data type of the day of week columns to int
df["whois_created_dayofweek"] =
df["whois_created_dayofweek"].astype("int64")
df["ctlog_earliest_dayofweek"] =
df["ctlog_earliest_dayofweek"].astype("int64")
df["ctlog_latest_dayofweek"] = df["ctlog_latest_dayofweek"].astype("int64")

# domain to cert delta
df["domain_to_earliest_cert_delta"] = df.apply(
        lambda x: utils.get_days_delta(
```

```
            x["ctlog_earliest"],
            x["whois_created"]
            if x["whois_created"] and x["ctlog_earliest"]
            else None,
        ),
        axis=1,
    )

# set the data type of the domain_to_cert_delta column to float
df["domain_to_earliest_cert_delta"] =
df["domain_to_earliest_cert_delta"].astype("float64")

# domain to latest cert delta
df["domain_to_latest_cert_delta"] = df.apply(
        lambda x: utils.get_days_delta(
            x["ctlog_latest"],
            x["whois_created"]
            if x["whois_created"] and x["ctlog_latest"]
            else None,
        ),
        axis=1,
    )

# set the data type of the domain_to_cert_delta column to float
df["domain_to_latest_cert_delta"] =
df["domain_to_latest_cert_delta"].astype("float64")

# drop columns we imported that we will never use
df = df.drop(columns=["url", "verification_time", "dateadded",
"soa_timestamp","domain_to_cert_delta"])

click.echo(df.head())
click.echo(df.describe(include='all'))
click.echo(df.dtypes)
                            domain  malicious       whois_created
0  i-db5p-cor001.api.p001.1drv.com      False 2013-08-05 18:33:50  \
4       soundcloud-pax.pandora.com      False 1993-12-28 05:00:00
5         joolcomercializadora.com       True 2023-05-22 14:53:50
6        createpdf-asr.acrobat.com      False 1999-03-16 05:00:00
8                          popt.in      False 2016-05-14 16:58:55

       ctlog_earliest        ctlog_latest  ctlog_wildcard
0 2022-01-24 20:01:58 2023-06-08 20:46:06            True  \
4 2022-05-19 00:00:00 2023-06-19 23:59:59            True
5 2022-04-06 22:23:24 2023-09-22 23:59:59           False
6 2022-09-09 00:00:00 2023-10-10 23:59:59            True
8 2023-01-07 20:36:15 2023-08-15 04:16:52           False
```

```
   whois_created_dayofweek  ctlog_earliest_dayofweek
ctlog_latest_dayofweek
0                        0                         0
3  \
4                        1                         3
0
5                        0                         2
4
6                        1                         4
1
8                        5                         5
1

   domain_to_earliest_cert_delta  domain_to_latest_cert_delta
0                        -3095.0                      -3595.0
4                       -10369.0                     -10766.0
5                          410.0                       -124.0
6                        -8578.0                      -8975.0
8                        -2430.0                      -2649.0
                  domain malicious                  whois_created
count              21549     21549                          21549  \
unique             21536         2                            NaN
top     www.mediafire.com     False                           NaN
freq                   2     11739                            NaN
mean                 NaN       NaN  2012-10-03 12:56:32.335050496
min                  NaN       NaN            1986-01-09 00:00:00
25%                  NaN       NaN            2003-05-25 13:35:05
50%                  NaN       NaN            2015-05-07 23:56:05
75%                  NaN       NaN            2023-03-20 15:03:16
max                  NaN       NaN            2023-07-03 08:21:24
std                  NaN       NaN                            NaN

                       ctlog_earliest                   ctlog_latest
count                           21549                          21549  \
unique                            NaN                            NaN
top                               NaN                            NaN
freq                              NaN                            NaN
mean    2022-09-26 15:45:50.943570432  2023-08-14 17:49:06.400900352
min               2021-11-30 05:24:28            2023-01-01 18:42:11
25%               2022-06-24 13:47:12            2023-07-02 08:11:07
50%               2022-10-18 21:00:14            2023-08-21 21:40:11
75%               2022-12-14 00:00:00            2023-09-21 19:41:38
max               2023-06-28 04:36:22            2023-12-31 23:59:59
std                               NaN                            NaN

       ctlog_wildcard  whois_created_dayofweek  ctlog_earliest_dayofweek
count           21549             21549.000000              21549.000000  \
unique              2                      NaN                       NaN
top             False                      NaN                       NaN
```

```
freq            13032                   NaN                       NaN
mean              NaN              2.332823                  2.399462
min               NaN              0.000000                  0.000000
25%               NaN              1.000000                  1.000000
50%               NaN              2.000000                  2.000000
75%               NaN              4.000000                  4.000000
max               NaN              6.000000                  6.000000
std               NaN              1.775043                  1.897252

        ctlog_latest_dayofweek  domain_to_earliest_cert_delta
count             21549.000000                   21549.000000  \
unique                     NaN                            NaN
top                        NaN                            NaN
freq                       NaN                            NaN
mean                  2.873080                   -3645.602070
min                   0.000000                  -13445.000000
25%                   1.000000                   -7078.000000
50%                   3.000000                   -2637.000000
75%                   5.000000                      69.000000
max                   6.000000                     524.000000
std                   2.057394                    3790.677119

        domain_to_latest_cert_delta
count                  21549.000000
unique                          NaN
top                             NaN
freq                            NaN
mean                   -3967.678222
min                   -13798.000000
25%                    -7421.000000
50%                    -3009.000000
75%                     -144.000000
max                      135.000000
std                     3852.703681
domain                          string[python]
malicious                                 bool
whois_created                   datetime64[ns]
ctlog_earliest                  datetime64[ns]
ctlog_latest                    datetime64[ns]
ctlog_wildcard                            bool
whois_created_dayofweek                  int64
ctlog_earliest_dayofweek                 int64
ctlog_latest_dayofweek                   int64
domain_to_earliest_cert_delta          float64
domain_to_latest_cert_delta            float64
dtype: object
```

In [3]:

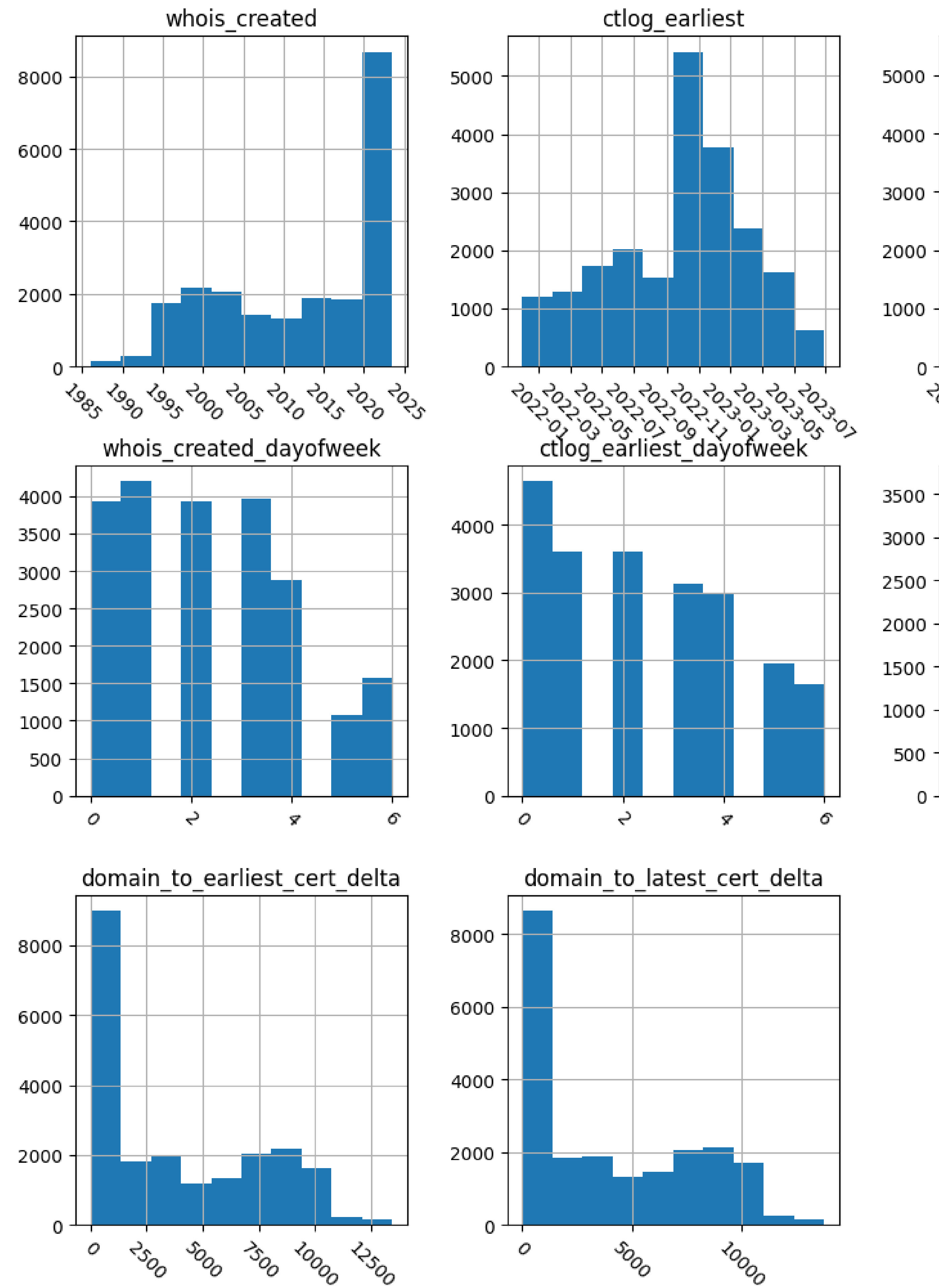
```
# absolute value of the domain_to_cert_delta
```

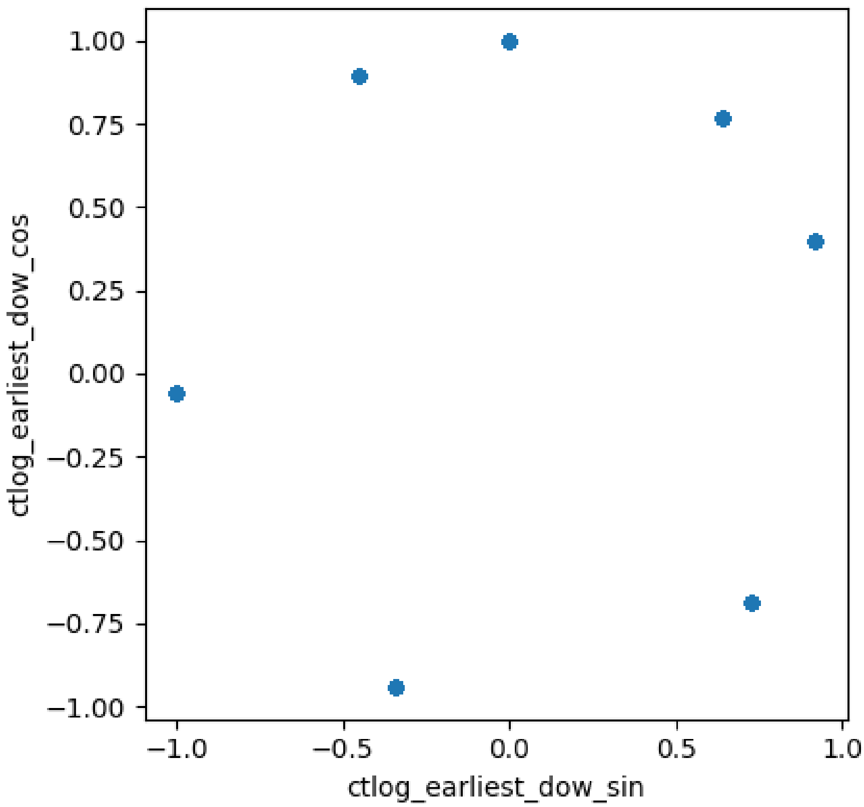
```
df["domain_to_earliest_cert_delta"] =
abs(df["domain_to_earliest_cert_delta"])
df["domain_to_latest_cert_delta"] = abs(df["domain_to_latest_cert_delta"])

df.hist(figsize=(12,12), xrot=-45)

col_index = df.columns.get_loc("whois_created_dayofweek")
df["whois_created_dow_sin"] = df.apply(lambda row:
math.sin(360/7*(row[col_index])),axis=1)
df["whois_created_dow_cos"] = df.apply(lambda row:
math.cos(360/7*(row[col_index])),axis=1)

col_index = df.columns.get_loc("ctlog_earliest_dayofweek")
df["ctlog_earliest_dow_sin"] = df.apply(lambda row:
math.sin(360/7*(row[col_index])),axis=1)
df["ctlog_earliest_dow_cos"] = df.apply(lambda row:
math.cos(360/7*(row[col_index])),axis=1)

col_index = df.columns.get_loc("ctlog_latest_dayofweek")
df["ctlog_latest_dow_sin"] = df.apply(lambda row:
math.sin(360/7*(row[col_index])),axis=1)
df["ctlog_latest_dow_cos"] = df.apply(lambda row:
math.cos(360/7*(row[col_index])),axis=1)


df.plot.scatter(x="ctlog_earliest_dow_sin",
y="ctlog_earliest_dow_cos").set_aspect('equal')
df.plot.scatter(x="whois_created_dow_sin",
y="whois_created_dow_cos").set_aspect('equal')
df.plot.scatter(x="ctlog_latest_dow_sin",
y="ctlog_latest_dow_cos").set_aspect('equal')

"""
# add one hot encode ctlog_earliest_not_before_dayofweek
df = pd.get_dummies(
        df,
        columns=["ctlog_earliest_dayofweek"],
        prefix=["ctlog_earliest_dow"],
    )
# add one hot encode whois_created_dayofweek to columns
df = pd.get_dummies(
        df, columns=["whois_created_dayofweek"], prefix="whois_dow"
    )
    """

# Summary statistics
click.echo(df.describe(include='all'))
                  domain malicious                  whois_created
count              21549     21549                          21549  \
```

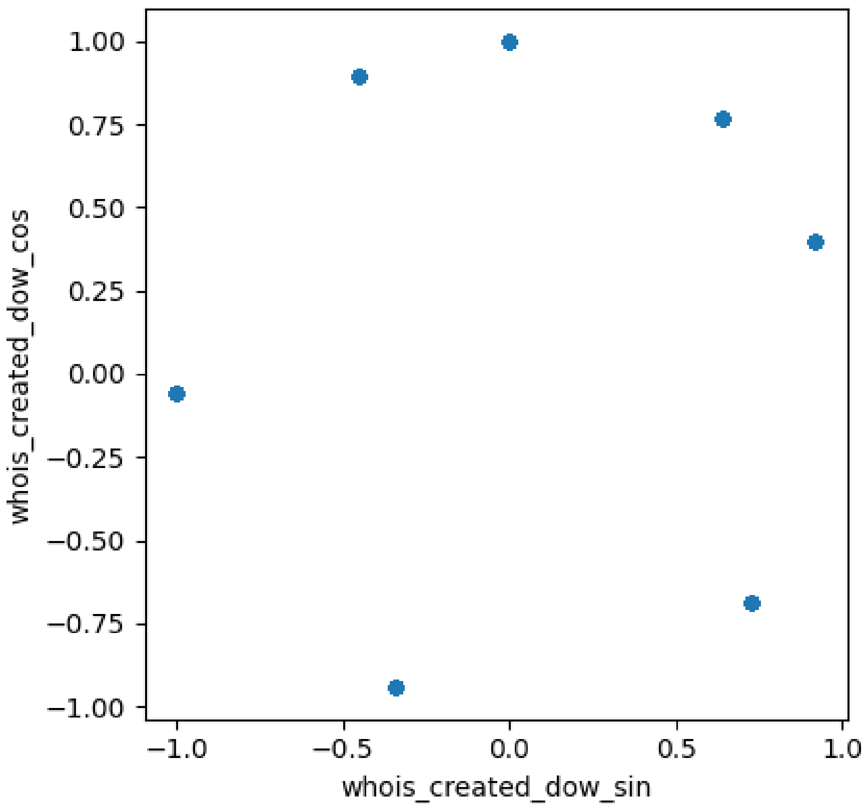
```
unique           21536         2                              NaN
top     www.mediafire.com     False                              NaN
freq                 2     11739                              NaN
mean               NaN       NaN  2012-10-03 12:56:32.335050496
min                NaN       NaN            1986-01-09 00:00:00
25%                NaN       NaN            2003-05-25 13:35:05
50%                NaN       NaN            2015-05-07 23:56:05
75%                NaN       NaN            2023-03-20 15:03:16
max                NaN       NaN            2023-07-03 08:21:24
std                NaN       NaN                              NaN

                     ctlog_earliest                   ctlog_latest
count                         21549                          21549  \
unique                          NaN                            NaN
top                             NaN                            NaN
freq                            NaN                            NaN
mean    2022-09-26 15:45:50.943570432  2023-08-14 17:49:06.400900352
min               2021-11-30 05:24:28            2023-01-01 18:42:11
25%               2022-06-24 13:47:12            2023-07-02 08:11:07
50%               2022-10-18 21:00:14            2023-08-21 21:40:11
75%               2022-12-14 00:00:00            2023-09-21 19:41:38
max               2023-06-28 04:36:22            2023-12-31 23:59:59
std                             NaN                            NaN

       ctlog_wildcard  whois_created_dayofweek  ctlog_earliest_dayofweek
count           21549             21549.000000              21549.000000  \
unique              2                      NaN                       NaN
top             False                      NaN                       NaN
freq            13032                      NaN                       NaN
mean              NaN                 2.332823                  2.399462
min               NaN                 0.000000                  0.000000
25%               NaN                 1.000000                  1.000000
50%               NaN                 2.000000                  2.000000
75%               NaN                 4.000000                  4.000000
max               NaN                 6.000000                  6.000000
std               NaN                 1.775043                  1.897252

        ctlog_latest_dayofweek  domain_to_earliest_cert_delta
count             21549.000000                   21549.000000  \
unique                     NaN                            NaN
top                        NaN                            NaN
freq                       NaN                            NaN
mean                  2.873080                    3742.948397
min                   0.000000                       0.000000
25%                   1.000000                     181.000000
50%                   3.000000                    2637.000000
75%                   5.000000                    7078.000000
max                   6.000000                   13445.000000
std                   2.057394                    3694.584062
```

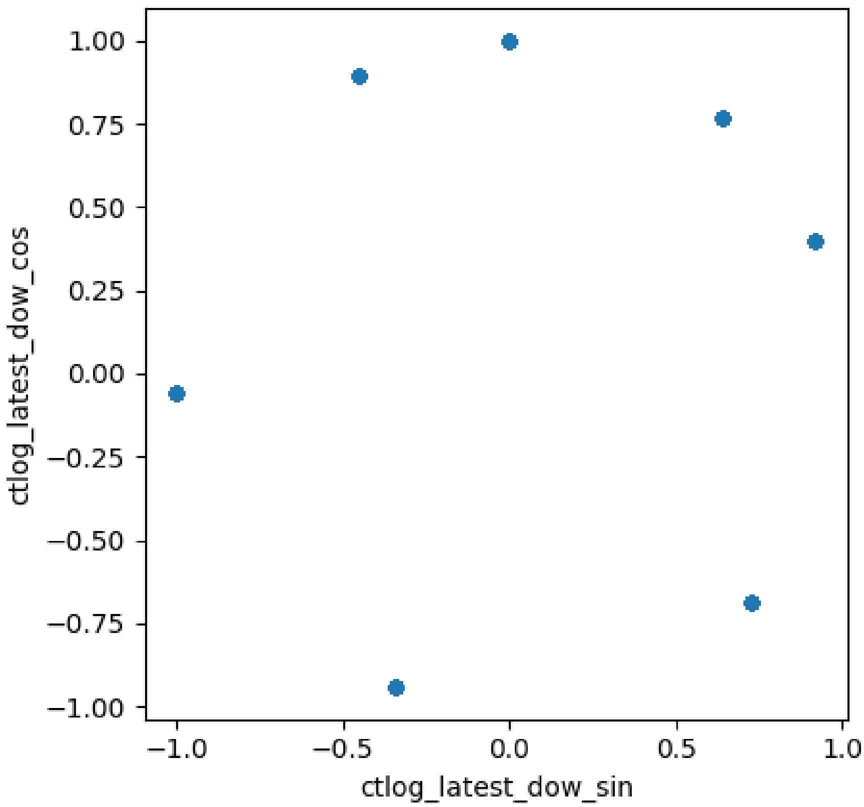
```
        domain_to_latest_cert_delta  whois_created_dow_sin
count                  21549.000000           21549.000000  \
unique                          NaN                    NaN
top                             NaN                    NaN
freq                            NaN                    NaN
mean                    3969.491206               0.140419
min                        0.000000              -0.998199
25%                      144.000000              -0.340712
50%                     3009.000000               0.000000
75%                     7421.000000               0.728010
max                    13798.000000               0.918032
std                     3850.835626               0.659922

        whois_created_dow_cos  ctlog_earliest_dow_sin
ctlog_earliest_dow_cos
count            21549.000000            21549.000000
21549.000000  \
unique                    NaN                     NaN
NaN
top                       NaN                     NaN
NaN
freq                      NaN                     NaN
NaN
mean                 0.054288                0.095357
0.161451
min                 -0.940168               -0.998199               -
0.940168
25%                 -0.685567               -0.340712               -
0.685567
50%                  0.396506                0.000000
0.396506
75%                  0.767830                0.728010
0.892589
max                  1.000000                0.918032
1.000000
std                  0.736128                0.651782
0.734891

        ctlog_latest_dow_sin  ctlog_latest_dow_cos
count           21549.000000          21549.000000
unique                   NaN                   NaN
top                      NaN                   NaN
freq                     NaN                   NaN
mean                0.096253              0.255578
min                -0.998199             -0.940168
25%                -0.450871             -0.685567
50%                 0.000000              0.396506
75%                 0.728010              0.892589
```

```
max                 0.918032            1.000000
std                 0.651597            0.707728
```


whois_created
ctlog_earliest
whois_created_dayofweek
ctlog_earliest_dayofweek
domain_to_earliest_cert_delta
domain_to_latest_cert_delta


ctlog_earliest_dow_cos
ctlog_earliest_dow_sin

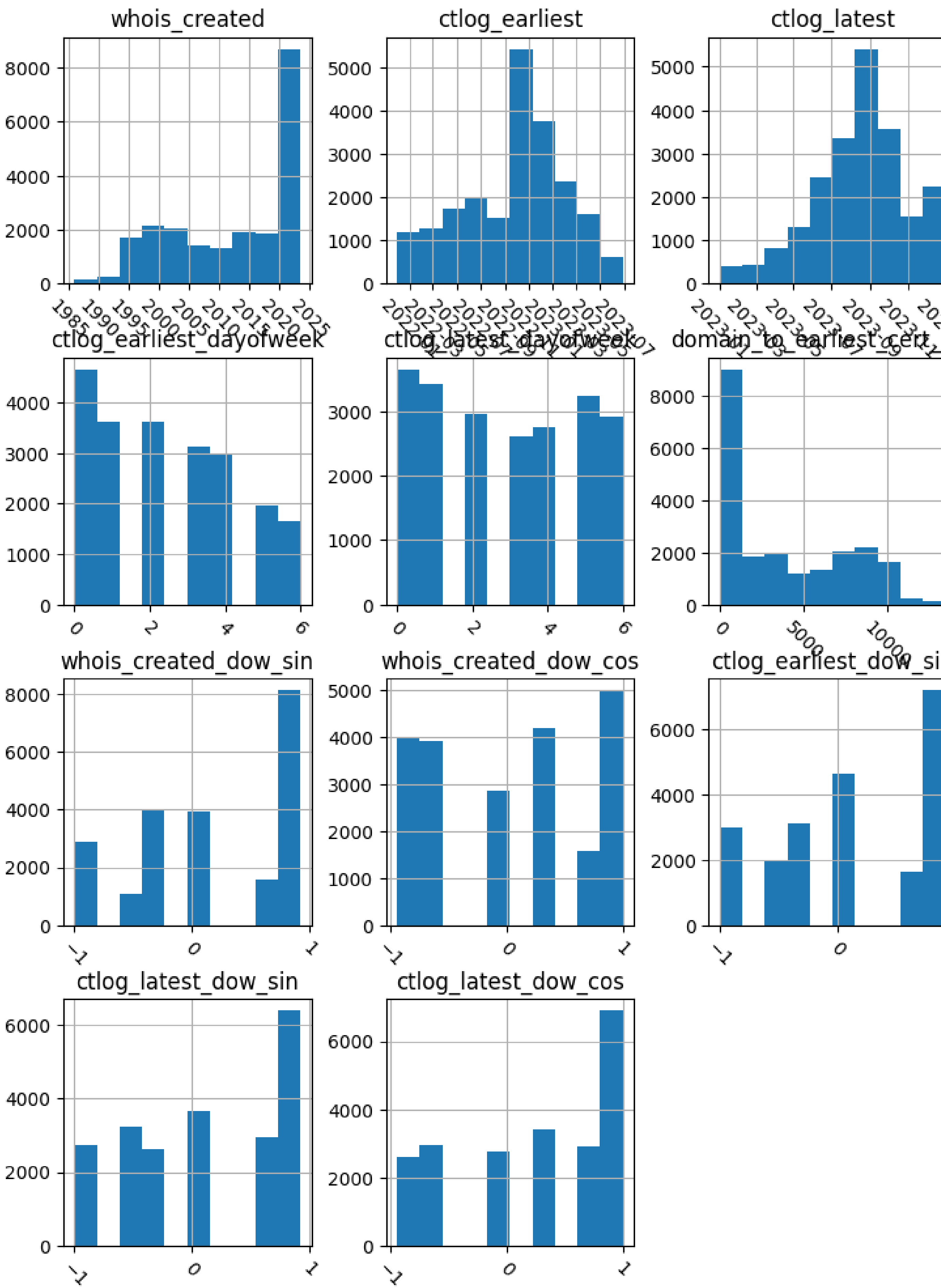

whois_created_dow_cos
whois_created_dow_sin

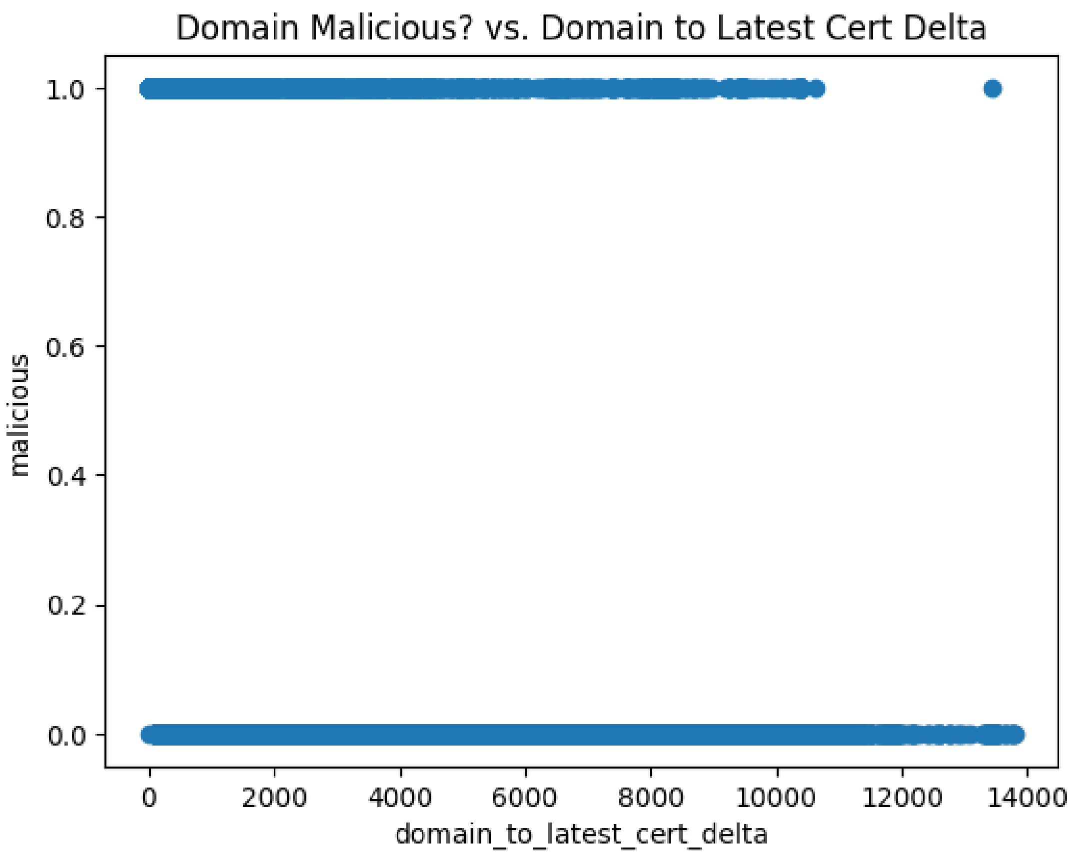


In [4]:

```
# Plot histograms
df.hist(figsize=(12,12), xrot=-45)

# Create a scatter plot of the data, showing malicious and benign domains
# plot the data as a scatter plot
plt.scatter(df["domain_to_earliest_cert_delta"], df["malicious"])
plt.xlabel("domain_to_earliest_cert_delta")
plt.ylabel("malicious")
plt.title("Domain Malicious? vs. Domain to Earliest Cert Delta")
plt.show()
# and for the latest
plt.scatter(df["domain_to_latest_cert_delta"], df["malicious"])
plt.xlabel("domain_to_latest_cert_delta")
plt.ylabel("malicious")
plt.title("Domain Malicious? vs. Domain to Latest Cert Delta")
plt.show()

click.echo(df.head())

# print a count of malicious and benign values
click.echo(df["malicious"].value_counts())
# deduplicate any rows, there shouldn't be any, but just in case
df = df.drop_duplicates()
click.echo(df["malicious"].value_counts())

click.echo(df.head())
```

```
X = df.drop(["malicious","domain", "ctlog_earliest", "ctlog_latest",
"whois_created", "whois_created_dayofweek", "ctlog_earliest_dayofweek"],
axis=1)
X = X.filter(items=combo_features)
y = df.filter(["malicious"])

click.echo("-------------------------------------")
click.echo(X.describe())
```


whois_created
ctlog_earliest
ctlog_latest
ctlog_earliest_dayofweek
ctlog_latest_dayofweek
domain_to_earliest_cert_
whois_created_dow_sin
whois_created_dow_cos
ctlog_earliest_dow_si
ctlog_latest_dow_sin
ctlog_latest_dow_cos




```
                              domain  malicious       whois_created
0  i-db5p-cor001.api.p001.1drv.com      False 2013-08-05 18:33:50   \
4       soundcloud-pax.pandora.com      False 1993-12-28 05:00:00
5         joolcomercializadora.com       True 2023-05-22 14:53:50
6        createpdf-asr.acrobat.com      False 1999-03-16 05:00:00
8                          popt.in      False 2016-05-14 16:58:55

       ctlog_earliest         ctlog_latest  ctlog_wildcard
0 2022-01-24 20:01:58 2023-06-08 20:46:06            True   \
4 2022-05-19 00:00:00 2023-06-19 23:59:59            True
5 2022-04-06 22:23:24 2023-09-22 23:59:59           False
6 2022-09-09 00:00:00 2023-10-10 23:59:59            True
8 2023-01-07 20:36:15 2023-08-15 04:16:52           False

   whois_created_dayofweek  ctlog_earliest_dayofweek
ctlog_latest_dayofweek
0                        0                          0
3  \
4                        1                          3
0
5                        0                          2
4
6                        1                          4
1
8                        5                          5
1
```

```
   domain_to_earliest_cert_delta  domain_to_latest_cert_delta
0                         3095.0                       3595.0  \
4                        10369.0                      10766.0
5                          410.0                        124.0
6                         8578.0                       8975.0
8                         2430.0                       2649.0

   whois_created_dow_sin  whois_created_dow_cos  ctlog_earliest_dow_sin
0               0.000000               1.000000               0.000000  \
4               0.918032               0.396506              -0.340712
5               0.000000               1.000000               0.728010
6               0.918032               0.396506              -0.998199
8              -0.450871               0.892589              -0.450871

   ctlog_earliest_dow_cos  ctlog_latest_dow_sin  ctlog_latest_dow_cos
0                1.000000             -0.340712             -0.940168
4               -0.940168              0.000000              1.000000
5               -0.685567             -0.998199             -0.059997
6               -0.059997              0.918032              0.396506
8                0.892589              0.918032              0.396506
malicious
False    11739
True      9810
Name: count, dtype: int64
malicious
False    11739
True      9810
Name: count, dtype: int64
                            domain  malicious       whois_created
0  i-db5p-cor001.api.p001.1drv.com      False 2013-08-05 18:33:50  \
4       soundcloud-pax.pandora.com      False 1993-12-28 05:00:00
5         joolcomercializadora.com       True 2023-05-22 14:53:50
6        createpdf-asr.acrobat.com      False 1999-03-16 05:00:00
8                          popt.in      False 2016-05-14 16:58:55

       ctlog_earliest        ctlog_latest  ctlog_wildcard
0 2022-01-24 20:01:58 2023-06-08 20:46:06            True  \
4 2022-05-19 00:00:00 2023-06-19 23:59:59            True
5 2022-04-06 22:23:24 2023-09-22 23:59:59           False
6 2022-09-09 00:00:00 2023-10-10 23:59:59            True
8 2023-01-07 20:36:15 2023-08-15 04:16:52           False

   whois_created_dayofweek  ctlog_earliest_dayofweek
ctlog_latest_dayofweek
0                        0                         0
3  \
4                        1                         3
0
```

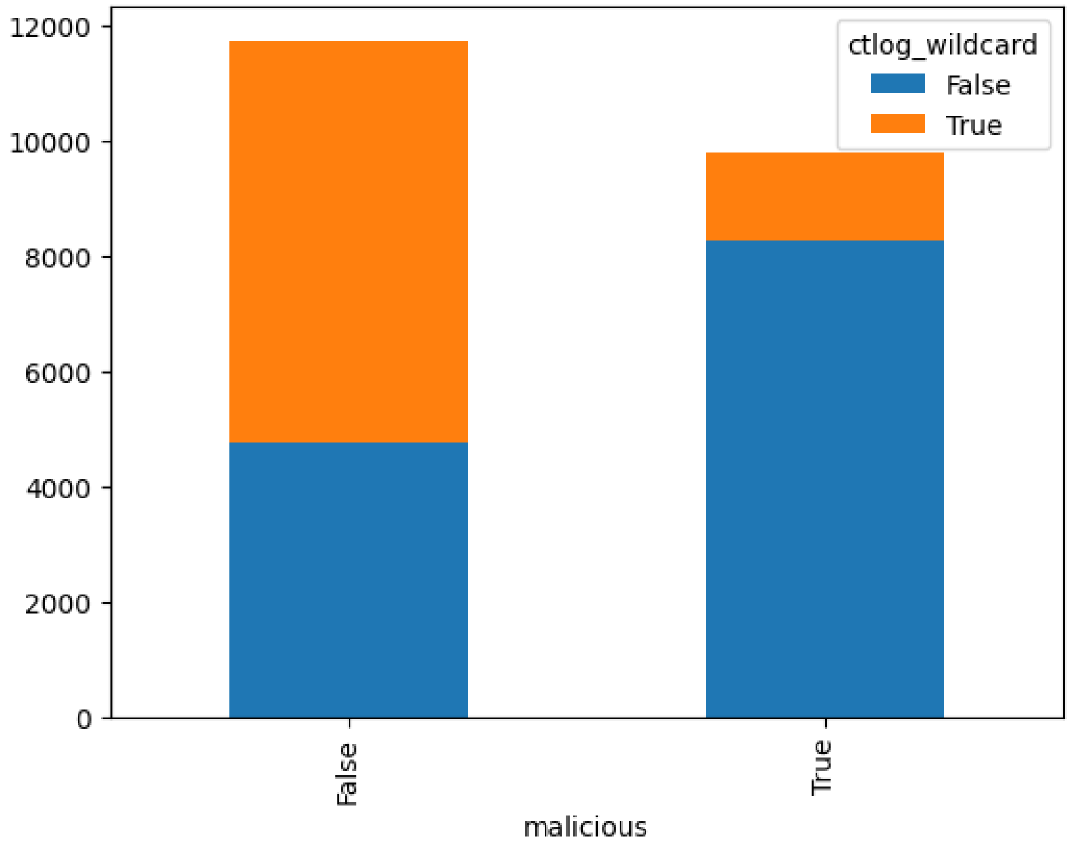
```
5                    0                    2
4
6                    1                    4
1
8                    5                    5
1

   domain_to_earliest_cert_delta  domain_to_latest_cert_delta
0                         3095.0                       3595.0  \
4                        10369.0                      10766.0
5                          410.0                        124.0
6                         8578.0                       8975.0
8                         2430.0                       2649.0

   whois_created_dow_sin  whois_created_dow_cos  ctlog_earliest_dow_sin
0               0.000000               1.000000                0.000000  \
4               0.918032               0.396506               -0.340712
5               0.000000               1.000000                0.728010
6               0.918032               0.396506               -0.998199
8              -0.450871               0.892589               -0.450871

   ctlog_earliest_dow_cos  ctlog_latest_dow_sin  ctlog_latest_dow_cos
0                1.000000             -0.340712             -0.940168
4               -0.940168              0.000000              1.000000
5               -0.685567             -0.998199             -0.059997
6               -0.059997              0.918032              0.396506
8                0.892589              0.918032              0.396506
-------------------------------------
       domain_to_earliest_cert_delta  ctlog_earliest_dow_sin
count                   21549.000000            21549.000000  \
mean                     3742.948397                0.095357
std                      3694.584062                0.651782
min                         0.000000               -0.998199
25%                       181.000000               -0.340712
50%                      2637.000000                0.000000
75%                      7078.000000                0.728010
max                     13445.000000                0.918032

       ctlog_earliest_dow_cos  whois_created_dow_sin  whois_created_dow_cos
count            21549.000000           21549.000000           21549.000000
\
mean                 0.161451               0.140419               0.054288
std                  0.734891               0.659922               0.736128
min                 -0.940168              -0.998199              -0.940168
25%                 -0.685567              -0.340712              -0.685567
50%                  0.396506               0.000000               0.396506
75%                  0.892589               0.728010               0.767830
max                  1.000000               0.918032               1.000000
```

```
       domain_to_latest_cert_delta  ctlog_latest_dow_sin
ctlog_latest_dow_cos
count                 21549.000000         21549.000000
21549.000000
mean                   3969.491206             0.096253
0.255578
std                    3850.835626             0.651597
0.707728
min                       0.000000            -0.998199             -
0.940168
25%                     144.000000            -0.450871             -
0.685567
50%                    3009.000000             0.000000
0.396506
75%                    7421.000000             0.728010
0.892589
max                   13798.000000             0.918032
1.000000
```

In [5]:

```
# print a count where malicious and ctlog_wildcard is true or false
click.echo("Malicious True ctlog_wildcard value counts")
click.echo(df.loc[df["malicious"] ==
True]["ctlog_wildcard"].value_counts())
# print a count where benign and ctlog_wildcard is true or false
click.echo("Malicious False ctlog_wildcard value counts")
click.echo(df.loc[df["malicious"] ==
False]["ctlog_wildcard"].value_counts())
# output to a graph
df.groupby(["malicious",
"ctlog_wildcard"]).size().unstack().plot(kind="bar", stacked=True)
plt.show()
```

```
Malicious True ctlog_wildcard value counts
ctlog_wildcard
False    8271
True     1539
Name: count, dtype: int64
Malicious False ctlog_wildcard value counts
ctlog_wildcard
True     6978
False    4761
Name: count, dtype: int64
```



In [6]:

```
# convert y (malicious) to 1/0 int
y = y.astype('int')
# convert X["ctlog_wildcard"]  to 1/0 int
if "ctlog_wildcard" in X.columns:
    X["ctlog_wildcard"] = X["ctlog_wildcard"].astype('int')

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

X_train = sm.add_constant(X_train)
X_test = sm.add_constant(X_test)

smfit = sm.Logit(y_train,X_train).fit()

smfit.summary()
Optimization terminated successfully.
         Current function value: 0.285320
         Iterations 7
```

Out[6]:

Logit Regression Results

Dep. Variable:	malicious	**No. Observations:**	17239
Model:	Logit	**Df Residuals:**	17229
Method:	MLE	**Df Model:**	9
Date:	Tue, 08 Aug 2023	**Pseudo R-squ.:**	0.5862
Time:	19:11:32	**Log-Likelihood:**	-4918.6

converged:	True	**LL-Null:**	-11885.
Covariance Type:	nonrobust	**LLR p-value:**	0.000

	coef	**std err**	**z**	**P>|z|**	**[0.025**	**0.975]**
const	3.0653	0.055	55.621	0.000	2.957	3.173
domain_to_earliest_cert_delta	0.0077	0.000	38.972	0.000	0.007	0.008
ctlog_earliest_dow_sin	0.1150	0.040	2.856	0.004	0.036	0.194
ctlog_earliest_dow_cos	-0.3271	0.036	-9.161	0.000	-0.397	-0.257
ctlog_wildcard	-0.1616	0.058	-2.784	0.005	-0.275	-0.048
whois_created_dow_sin	0.0485	0.039	1.244	0.213	-0.028	0.125
whois_created_dow_cos	-0.0503	0.036	-1.412	0.158	-0.120	0.020
domain_to_latest_cert_delta	-0.0082	0.000	-41.438	0.000	-0.009	-0.008
ctlog_latest_dow_sin	-0.1573	0.040	-3.939	0.000	-0.236	-0.079
ctlog_latest_dow_cos	0.2594	0.038	6.904	0.000	0.186	0.333

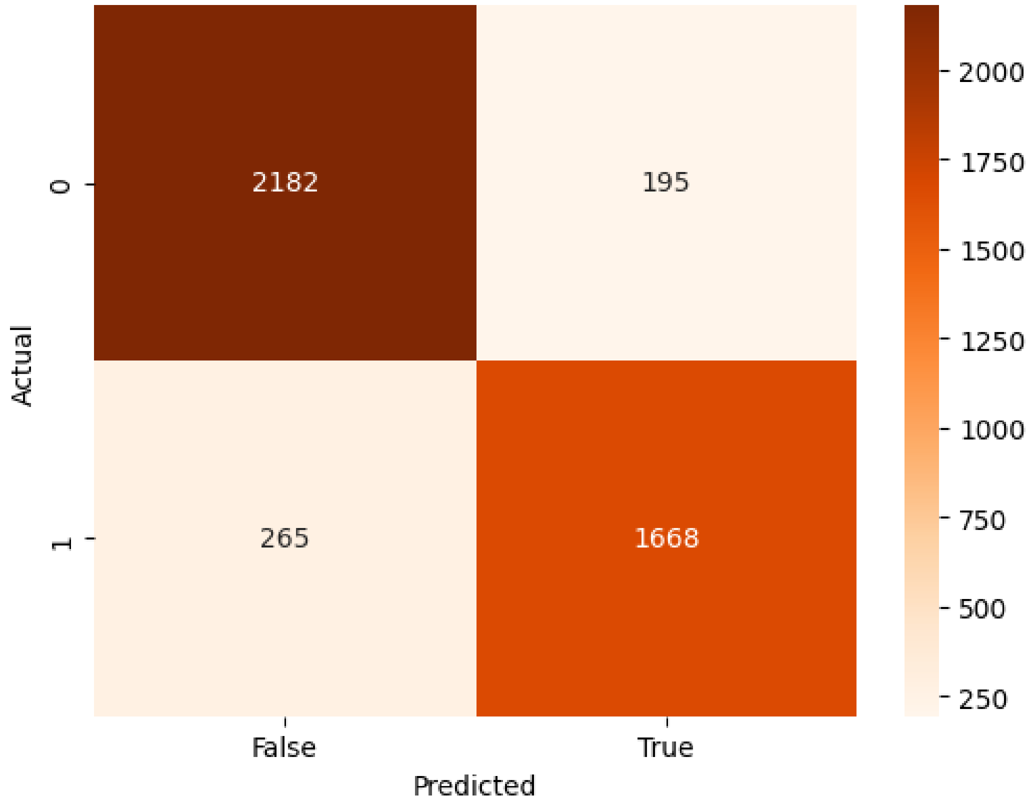
In [7]:

```
# Predict the malicious column using the test data
#add the incepts


y_predicted = smfit.predict(X_test)

# Present the results in a confusion matrix
confusion_matrix = confusion_matrix(y_test, y_predicted.round())
click.echo(confusion_matrix)

click.echo("Classification report:")
click.echo(classification_report(y_test, y_predicted.round()))

# Heatmap of confusion matrix
y_predicted

threshold = 0.5
y_predicted = [y > threshold for y in y_predicted]
data = {'Actual':    y_test.values.flatten(),
        'Predicted': y_predicted
        }

# Generate a confusion matrix and heatmap to evaluate the Type I and Type
II errors/ FP/FN etc.
df = pd.DataFrame(data, columns=['Actual','Predicted'])
confusion_matrix = pd.crosstab(df['Actual'], df['Predicted'],
rownames=['Actual'], colnames=['Predicted'])
fig = sns.heatmap(confusion_matrix, annot=True, cmap='Oranges', fmt='g')
fig
[[2182  195]
 [ 265 1668]]
Classification report:
              precision    recall  f1-score   support
```

```
           0       0.89      0.92      0.90      2377
           1       0.90      0.86      0.88      1933

    accuracy                           0.89      4310
   macro avg       0.89      0.89      0.89      4310
weighted avg       0.89      0.89      0.89      4310
```

Out[7]:

```
<Axes: xlabel='Predicted', ylabel='Actual'>
```

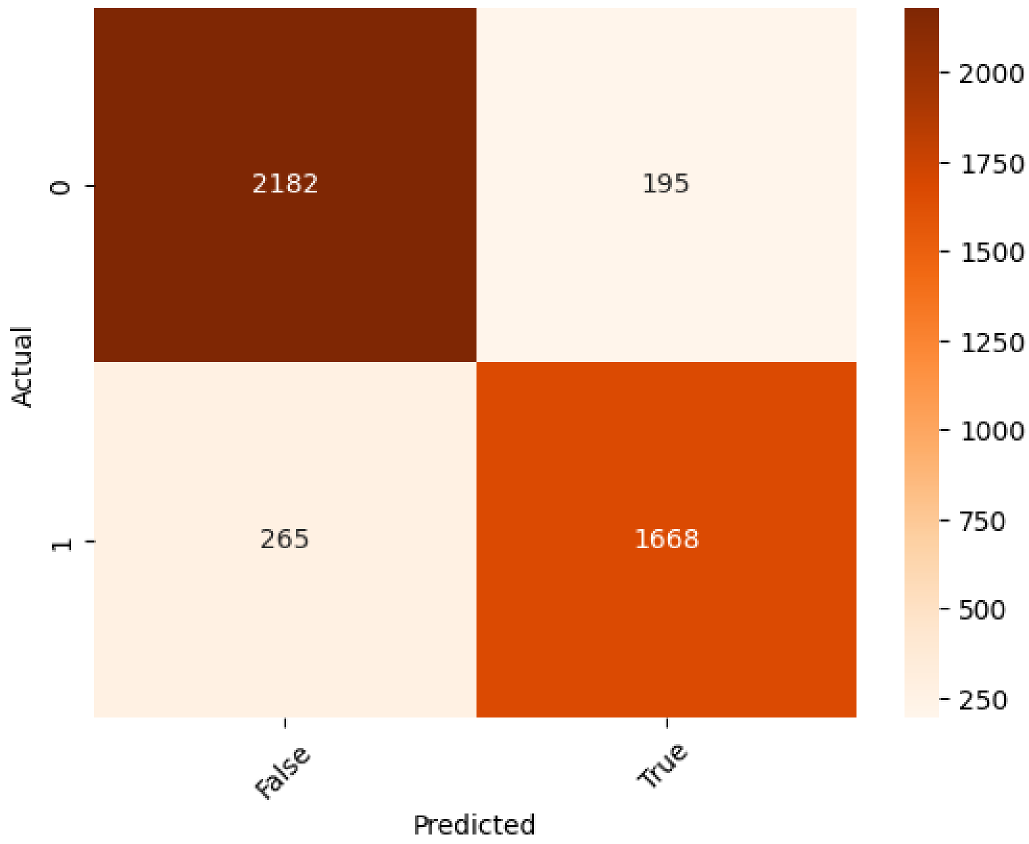


In [8]:

```
# rotate the confusion matrix
confusion_matrix = pd.crosstab(df['Actual'], df['Predicted'],
rownames=['Actual'], colnames=['Predicted'])
fig = sns.heatmap(confusion_matrix, annot=True, cmap='Oranges', fmt='g')
fig.set_xticklabels(fig.get_xticklabels(), rotation=45)
fig
```

Out[8]:

```
<Axes: xlabel='Predicted', ylabel='Actual'>
```


2182
195
265
1668
0
1
Actual
False
True
Predicted
2000
1750
1500
1250
1000
750
500
250

Appendix B: Rendered Jupyter Notebooks – Random Forest

I. Feature Set Baseline

In [1]:

```
import click
import pandas as pd
import glob
import os
#import sklearn
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.metrics import accuracy_score, f1_score, confusion_matrix,
classification_report, roc_auc_score, roc_curve, auc
from sklearn.preprocessing import StandardScaler
from scipy import stats
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
import statsmodels.api as sm
import matplotlib.pyplot as plt
from datetime import datetime, date
# qqplot of the data
import seaborn as sns
import math
import numpy as np
import utils

combo_features = ['domain_to_earliest_cert_delta']

path = "../data/"
filename = None

epoch = datetime(1970,1,1)
# open the training data file, from ../data/ for the most recent merged
training data file saved by prepare-training-data-parallel.py
full_path = path + "merged_20230705-104357_training_adorned-engineered.csv"

# get latest file matching the glob
if not filename:
    filename = max(glob.glob(full_path), key=os.path.getctime)
    click.echo(f"Using {filename} as training data")
    df = pd.read_csv(filename, sep=",")

# randomize the rows
df = df.sample(frac=1, random_state=42).reset_index(drop=True)

click.echo(df.shape)
click.echo(df.columns)

""" # From prepare-training-data-parallel.py:
# Columns:
```

```
url
verification_time
domain
malicious
dateadded
whois_created
soa_timestamp
ctlog_earliest
ctlog_latest
ctlog_wildcard
whois_created_dayofweek,
ctlog_earliest_dayofweek,
domain_to_cert_delta
"""


# Data cleaning - there may be some epoch values in the whois_created
#  & ct_log_earliest columns, so we need to remove those rows

# convert the whois_created and ctlog_earliest, ctlog_latest columns to
datetime

df = df.loc[df["whois_created"] != ""]
df = df.loc[df["ctlog_earliest"] != ""]
df = df.loc[df["ctlog_latest"] != ""]

# some dates are have nanoseconds in them, so we need to remove the
nanosecond part
df["whois_created"] = df["whois_created"].str.split(".", n = 1, expand =
True)[0]
# some dates has additional timezone information, so we need to remove that
df["whois_created"] = df["whois_created"].apply(lambda x: " ".join(
str(x).split(" ")[:2]))


# convert the whois_created and ct_log_earliest columns to datetime
df = df.astype({"whois_created": 'datetime64[ns]', "ctlog_earliest":
'datetime64[ns]', "ctlog_latest": 'datetime64[ns]'})
df = df.loc[df["whois_created"].ne(pd.Timestamp('1970-01-01 00:00:00'))]
df = df.loc[df["ctlog_earliest"].ne(pd.Timestamp('1970-01-01 00:00:00'))]
df = df.loc[df["ctlog_latest"].ne(pd.Timestamp('1970-01-01 00:00:00'))]


# malicious is a bool
df['malicious'] = df['malicious'].astype('bool')
df['domain'] = df['domain'].astype('string')
```

```
Using ../data/merged_20230705-104357_training_adorned-engineered.csv as
training data
(35438, 13)
```

```
Index(['url', 'verification_time', 'domain', 'malicious', 'dateadded',
       'whois_created', 'soa_timestamp', 'ctlog_earliest', 'ctlog_latest',
       'ctlog_wildcard', 'whois_created_dayofweek',
'ctlog_earliest_dayofweek',
       'domain_to_cert_delta'],
      dtype='object')
```

In [2]:

```
#########
# Feature engineering
#########

# remove any rows where the whois_created or ctlog_earliest columns are
null
df = df.loc[df["whois_created"].notnull()]
df = df.loc[df["ctlog_earliest"].notnull()]

# if the data is missing the "whois_created_dayofweek" or
"ctlog_earliest_dayofweek" columns, then add them
# whois created day of the week 0 = Monday, 6 = Sunday
df["whois_created_dayofweek"] = df["whois_created"].apply(
        lambda x: x.weekday() if isinstance(x, date) else None
    )

# cert valid day of the week 0 = Monday, 6 = Sunday
df["ctlog_earliest_dayofweek"] = df["ctlog_earliest"].apply(
        lambda x: x.weekday() if isinstance(x, date) else None
    )

df["ctlog_latest_dayofweek"] = df["ctlog_latest"].apply(
        lambda x: x.weekday() if isinstance(x, date) else None
    )

# set data type of the day of week columns to int
df["whois_created_dayofweek"] =
df["whois_created_dayofweek"].astype("int64")
df["ctlog_earliest_dayofweek"] =
df["ctlog_earliest_dayofweek"].astype("int64")
df["ctlog_latest_dayofweek"] = df["ctlog_latest_dayofweek"].astype("int64")

# domain to cert delta
df["domain_to_earliest_cert_delta"] = df.apply(
        lambda x: utils.get_days_delta(
            x["ctlog_earliest"],
            x["whois_created"]
            if x["whois_created"] and x["ctlog_earliest"]
            else None,
        ),
        axis=1,
    )
```

```
# set the data type of the domain_to_cert_delta column to float
df["domain_to_earliest_cert_delta"] =
df["domain_to_earliest_cert_delta"].astype("float64")

# domain to latest cert delta
df["domain_to_latest_cert_delta"] = df.apply(
        lambda x: utils.get_days_delta(
            x["ctlog_latest"],
            x["whois_created"]
            if x["whois_created"] and x["ctlog_latest"]
            else None,
        ),
        axis=1,
    )

# set the data type of the domain_to_cert_delta column to float
df["domain_to_latest_cert_delta"] =
df["domain_to_latest_cert_delta"].astype("float64")

# drop columns we imported that we will never use
df = df.drop(columns=["url", "verification_time", "dateadded",
"soa_timestamp","domain_to_cert_delta"])

click.echo(df.head())
click.echo(df.describe(include='all'))
click.echo(df.dtypes)
                            domain  malicious       whois_created
0  i-db5p-cor001.api.p001.1drv.com      False 2013-08-05 18:33:50  \
4       soundcloud-pax.pandora.com      False 1993-12-28 05:00:00
5         joolcomercializadora.com       True 2023-05-22 14:53:50
6        createpdf-asr.acrobat.com      False 1999-03-16 05:00:00
8                          popt.in      False 2016-05-14 16:58:55

       ctlog_earliest        ctlog_latest  ctlog_wildcard
0 2022-01-24 20:01:58 2023-06-08 20:46:06            True  \
4 2022-05-19 00:00:00 2023-06-19 23:59:59            True
5 2022-04-06 22:23:24 2023-09-22 23:59:59           False
6 2022-09-09 00:00:00 2023-10-10 23:59:59            True
8 2023-01-07 20:36:15 2023-08-15 04:16:52           False

   whois_created_dayofweek  ctlog_earliest_dayofweek
ctlog_latest_dayofweek
0                        0                          0
3  \
4                        1                          3
0
5                        0                          2
4
```

```
6                        1                         4
1
8                        5                         5
1

   domain_to_earliest_cert_delta  domain_to_latest_cert_delta
0                        -3095.0                      -3595.0
4                       -10369.0                     -10766.0
5                          410.0                       -124.0
6                        -8578.0                      -8975.0
8                        -2430.0                      -2649.0
                   domain malicious                  whois_created
count               21549    21549                          21549  \
unique              21536        2                            NaN
top      www.mediafire.com    False                            NaN
freq                    2    11739                            NaN
mean                  NaN      NaN  2012-10-03 12:56:32.335050496
min                   NaN      NaN            1986-01-09 00:00:00
25%                   NaN      NaN            2003-05-25 13:35:05
50%                   NaN      NaN            2015-05-07 23:56:05
75%                   NaN      NaN            2023-03-20 15:03:16
max                   NaN      NaN            2023-07-03 08:21:24
std                   NaN      NaN                            NaN

                       ctlog_earliest                   ctlog_latest
count                           21549                          21549  \
unique                            NaN                            NaN
top                               NaN                            NaN
freq                              NaN                            NaN
mean    2022-09-26 15:45:50.943570432  2023-08-14 17:49:06.400900352
min               2021-11-30 05:24:28            2023-01-01 18:42:11
25%               2022-06-24 13:47:12            2023-07-02 08:11:07
50%               2022-10-18 21:00:14            2023-08-21 21:40:11
75%               2022-12-14 00:00:00            2023-09-21 19:41:38
max               2023-06-28 04:36:22            2023-12-31 23:59:59
std                               NaN                            NaN

       ctlog_wildcard  whois_created_dayofweek  ctlog_earliest_dayofweek
count           21549             21549.000000              21549.000000  \
unique              2                      NaN                       NaN
top             False                      NaN                       NaN
freq            13032                      NaN                       NaN
mean              NaN                 2.332823                  2.399462
min               NaN                 0.000000                  0.000000
25%               NaN                 1.000000                  1.000000
50%               NaN                 2.000000                  2.000000
75%               NaN                 4.000000                  4.000000
max               NaN                 6.000000                  6.000000
std               NaN                 1.775043                  1.897252
```

```
       ctlog_latest_dayofweek  domain_to_earliest_cert_delta
count             21549.000000                   21549.000000  \
unique                     NaN                            NaN
top                        NaN                            NaN
freq                       NaN                            NaN
mean                  2.873080                   -3645.602070
min                   0.000000                  -13445.000000
25%                   1.000000                   -7078.000000
50%                   3.000000                   -2637.000000
75%                   5.000000                      69.000000
max                   6.000000                     524.000000
std                   2.057394                    3790.677119

       domain_to_latest_cert_delta
count                 21549.000000
unique                         NaN
top                            NaN
freq                           NaN
mean                  -3967.678222
min                  -13798.000000
25%                   -7421.000000
50%                   -3009.000000
75%                    -144.000000
max                     135.000000
std                    3852.703681
domain                              string[python]
malicious                                     bool
whois_created                       datetime64[ns]
ctlog_earliest                      datetime64[ns]
ctlog_latest                        datetime64[ns]
ctlog_wildcard                                bool
whois_created_dayofweek                      int64
ctlog_earliest_dayofweek                     int64
ctlog_latest_dayofweek                       int64
domain_to_earliest_cert_delta              float64
domain_to_latest_cert_delta                float64
dtype: object
```

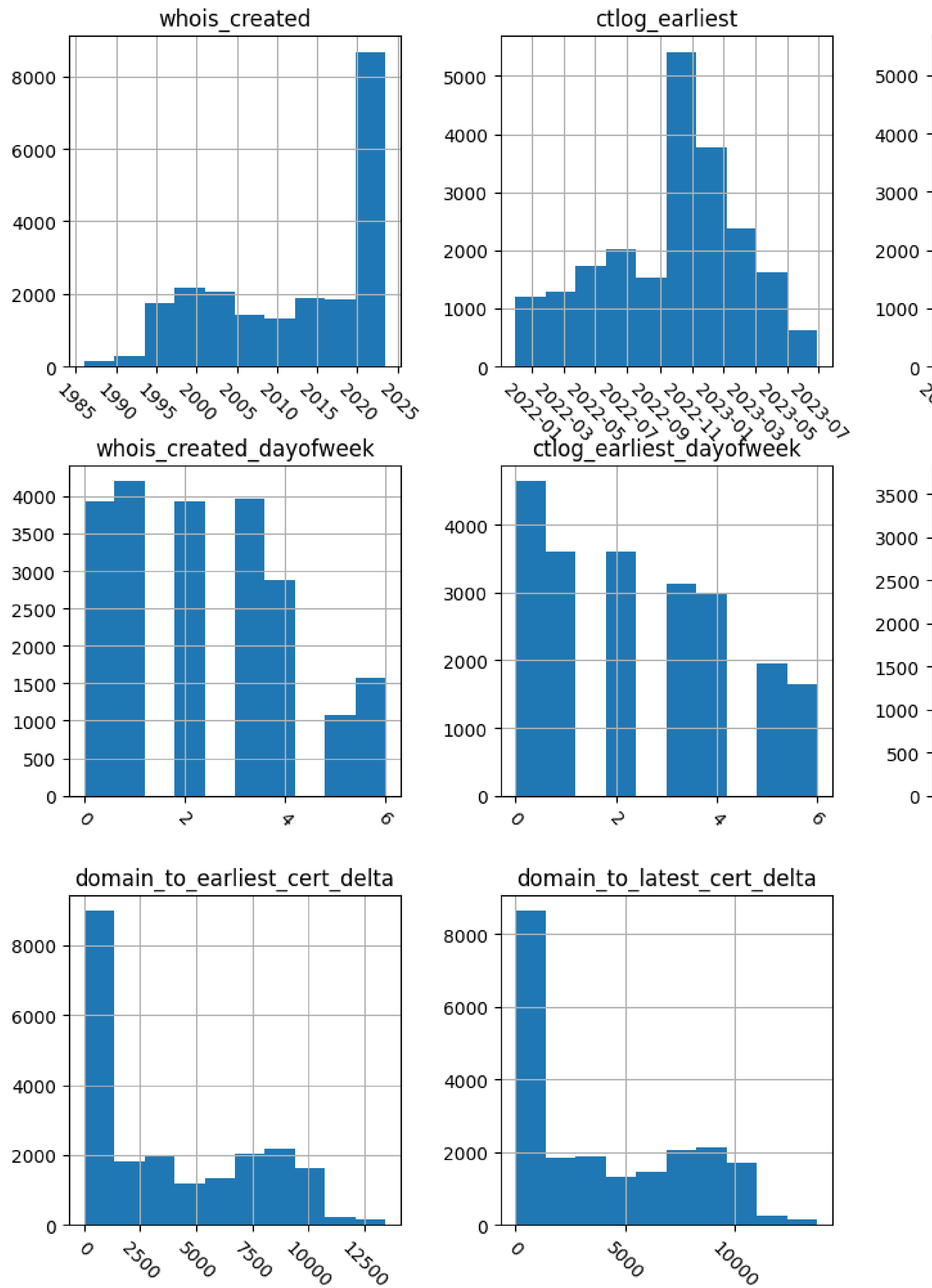
In [3]:

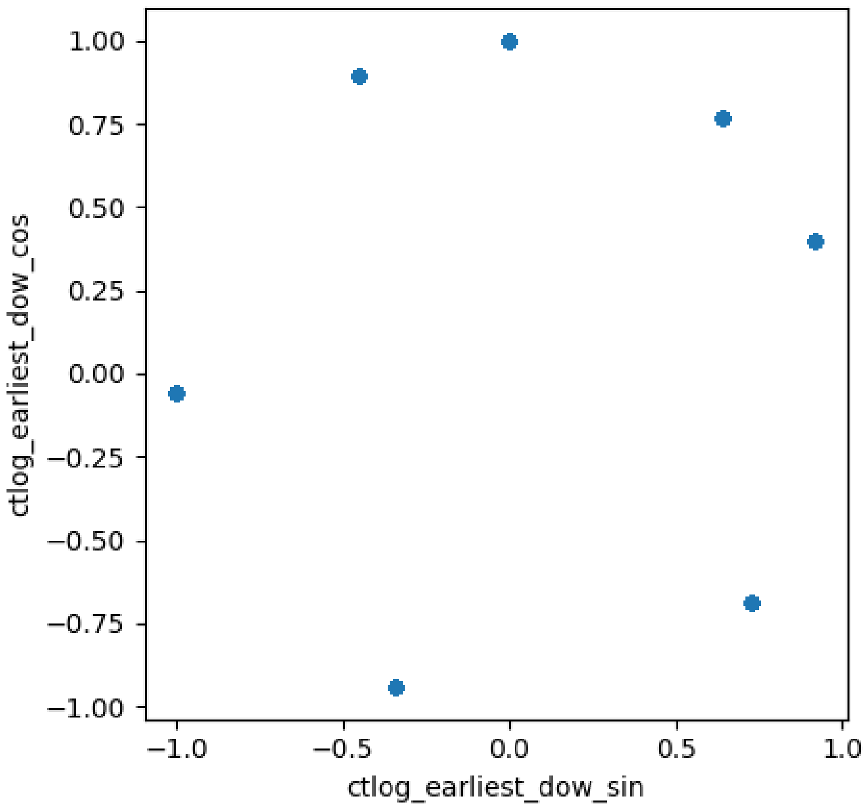
```
# absolute value of the domain_to_cert_delta
df["domain_to_earliest_cert_delta"] =
abs(df["domain_to_earliest_cert_delta"])
df["domain_to_latest_cert_delta"] = abs(df["domain_to_latest_cert_delta"])

df.hist(figsize=(12,12), xrot=-45)

col_index = df.columns.get_loc("whois_created_dayofweek")
df["whois_created_dow_sin"] = df.apply(lambda row:
math.sin(360/7*(row[col_index])),axis=1)
```

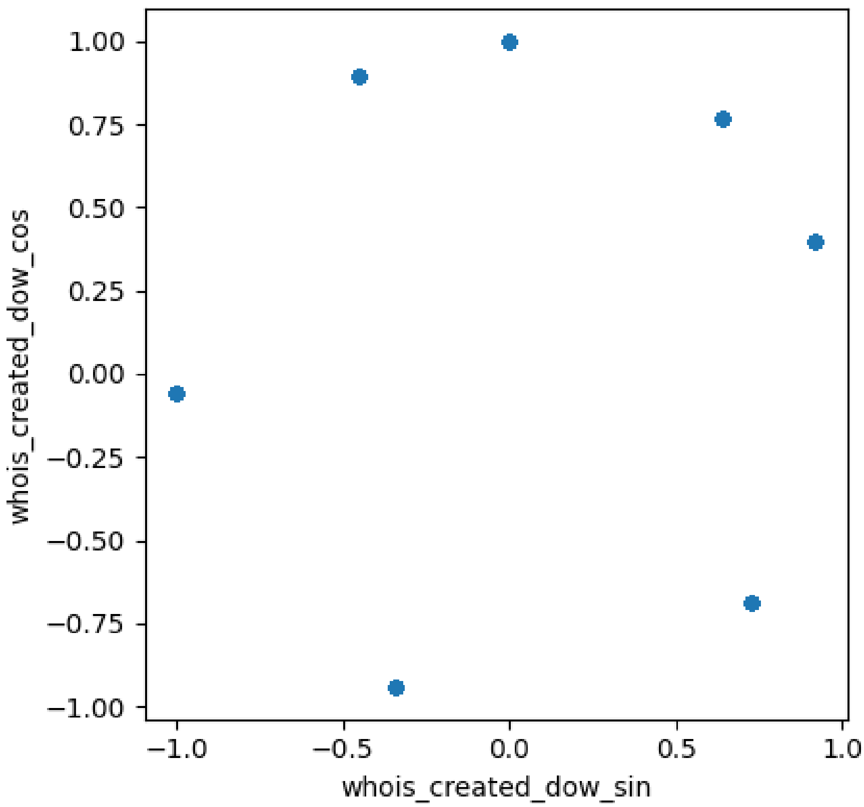
```
df["whois_created_dow_cos"] = df.apply(lambda row:
math.cos(360/7*(row[col_index])),axis=1)

col_index = df.columns.get_loc("ctlog_earliest_dayofweek")
df["ctlog_earliest_dow_sin"] = df.apply(lambda row:
math.sin(360/7*(row[col_index])),axis=1)
df["ctlog_earliest_dow_cos"] = df.apply(lambda row:
math.cos(360/7*(row[col_index])),axis=1)

col_index = df.columns.get_loc("ctlog_latest_dayofweek")
df["ctlog_latest_dow_sin"] = df.apply(lambda row:
math.sin(360/7*(row[col_index])),axis=1)
df["ctlog_latest_dow_cos"] = df.apply(lambda row:
math.cos(360/7*(row[col_index])),axis=1)


df.plot.scatter(x="ctlog_earliest_dow_sin",
y="ctlog_earliest_dow_cos").set_aspect('equal')
df.plot.scatter(x="whois_created_dow_sin",
y="whois_created_dow_cos").set_aspect('equal')
df.plot.scatter(x="ctlog_latest_dow_sin",
y="ctlog_latest_dow_cos").set_aspect('equal')

"""
# add one hot encode ctlog_earliest_not_before_dayofweek
df = pd.get_dummies(
        df,
        columns=["ctlog_earliest_dayofweek"],
        prefix=["ctlog_earliest_dow"],
    )
# add one hot encode whois_created_dayofweek to columns
df = pd.get_dummies(
        df, columns=["whois_created_dayofweek"], prefix="whois_dow"
    )
    """

# Summary statistics
click.echo(df.describe(include='all'))
                   domain malicious                   whois_created
count               21549     21549                           21549  \
unique              21536         2                             NaN
top     www.mediafire.com     False                             NaN
freq                    2     11739                             NaN
mean                  NaN       NaN  2012-10-03 12:56:32.335050496
min                   NaN       NaN            1986-01-09 00:00:00
25%                   NaN       NaN            2003-05-25 13:35:05
50%                   NaN       NaN            2015-05-07 23:56:05
75%                   NaN       NaN            2023-03-20 15:03:16
max                   NaN       NaN            2023-07-03 08:21:24
```

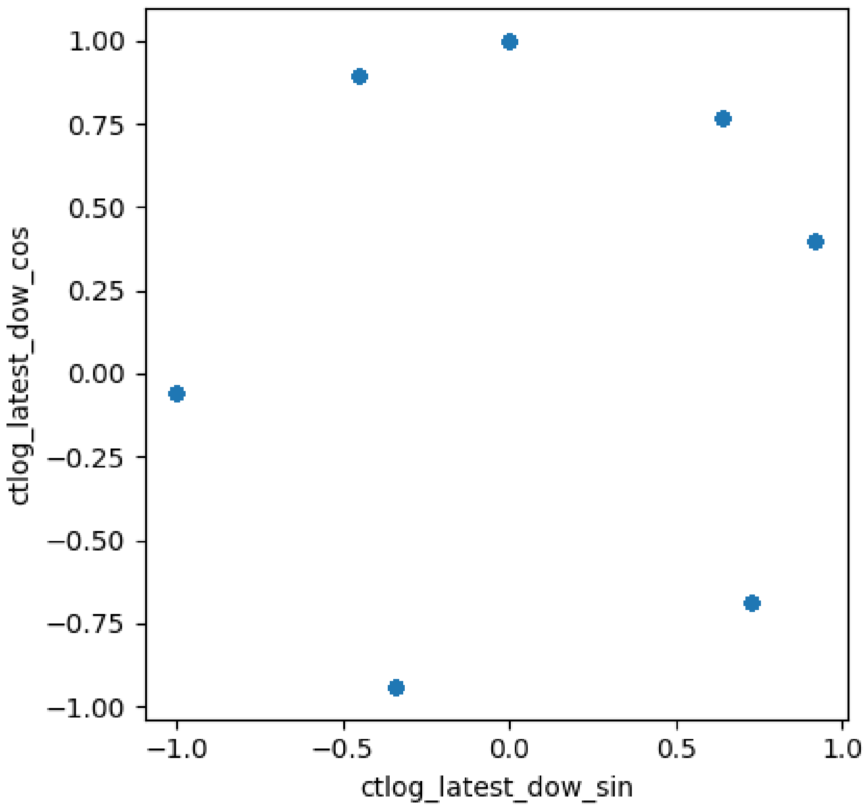
```
std                    NaN       NaN                            NaN

                     ctlog_earliest                  ctlog_latest
count                         21549                         21549  \
unique                          NaN                           NaN
top                             NaN                           NaN
freq                            NaN                           NaN
mean    2022-09-26 15:45:50.943570432  2023-08-14 17:49:06.400900352
min             2021-11-30 05:24:28           2023-01-01 18:42:11
25%             2022-06-24 13:47:12           2023-07-02 08:11:07
50%             2022-10-18 21:00:14           2023-08-21 21:40:11
75%             2022-12-14 00:00:00           2023-09-21 19:41:38
max             2023-06-28 04:36:22           2023-12-31 23:59:59
std                             NaN                           NaN

       ctlog_wildcard  whois_created_dayofweek  ctlog_earliest_dayofweek
count           21549             21549.000000              21549.000000  \
unique              2                      NaN                       NaN
top             False                      NaN                       NaN
freq            13032                      NaN                       NaN
mean              NaN                 2.332823                  2.399462
min               NaN                 0.000000                  0.000000
25%               NaN                 1.000000                  1.000000
50%               NaN                 2.000000                  2.000000
75%               NaN                 4.000000                  4.000000
max               NaN                 6.000000                  6.000000
std               NaN                 1.775043                  1.897252

        ctlog_latest_dayofweek  domain_to_earliest_cert_delta
count             21549.000000                   21549.000000  \
unique                     NaN                            NaN
top                        NaN                            NaN
freq                       NaN                            NaN
mean                  2.873080                    3742.948397
min                   0.000000                       0.000000
25%                   1.000000                     181.000000
50%                   3.000000                    2637.000000
75%                   5.000000                    7078.000000
max                   6.000000                   13445.000000
std                   2.057394                    3694.584062

        domain_to_latest_cert_delta  whois_created_dow_sin
count                  21549.000000           21549.000000  \
unique                          NaN                    NaN
top                             NaN                    NaN
freq                            NaN                    NaN
mean                    3969.491206               0.140419
min                        0.000000              -0.998199
25%                      144.000000              -0.340712
```

```
50%                     3009.000000                0.000000
75%                     7421.000000                0.728010
max                    13798.000000                0.918032
std                     3850.835626                0.659922

        whois_created_dow_cos  ctlog_earliest_dow_sin
ctlog_earliest_dow_cos
count            21549.000000            21549.000000
21549.000000  \
unique                    NaN                     NaN
NaN
top                       NaN                     NaN
NaN
freq                      NaN                     NaN
NaN
mean                 0.054288                0.095357
0.161451
min                 -0.940168               -0.998199               -
0.940168
25%                 -0.685567               -0.340712               -
0.685567
50%                  0.396506                0.000000
0.396506
75%                  0.767830                0.728010
0.892589
max                  1.000000                0.918032
1.000000
std                  0.736128                0.651782
0.734891

        ctlog_latest_dow_sin  ctlog_latest_dow_cos
count           21549.000000          21549.000000
unique                   NaN                   NaN
top                      NaN                   NaN
freq                     NaN                   NaN
mean                0.096253              0.255578
min                -0.998199             -0.940168
25%                -0.450871             -0.685567
50%                 0.000000              0.396506
75%                 0.728010              0.892589
max                 0.918032              1.000000
std                 0.651597              0.707728
```


whois_created
0
2000
4000
6000
8000
1985
1990
1995
2000
2005
2010
2015
2020
2025
ctlog_earliest
0
1000
2000
3000
4000
5000
2022-01
2022-03
2022-05
2022-07
2022-09
2022-11
2023-01
2023-03
2023-05
2023-07
whois_created_dayofweek
0
500
1000
1500
2000
2500
3000
3500
4000
0
2
4
6
ctlog_earliest_dayofweek
0
1000
2000
3000
4000
0
2
4
6
domain_to_earliest_cert_delta
0
2000
4000
6000
8000
0
2500
5000
7500
10000
12500
domain_to_latest_cert_delta
0
2000
4000
6000
8000
0
5000
10000


1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
ctlog_earliest_dow_cos
−1.0
−0.5
0.0
0.5
1.0
ctlog_earliest_dow_sin

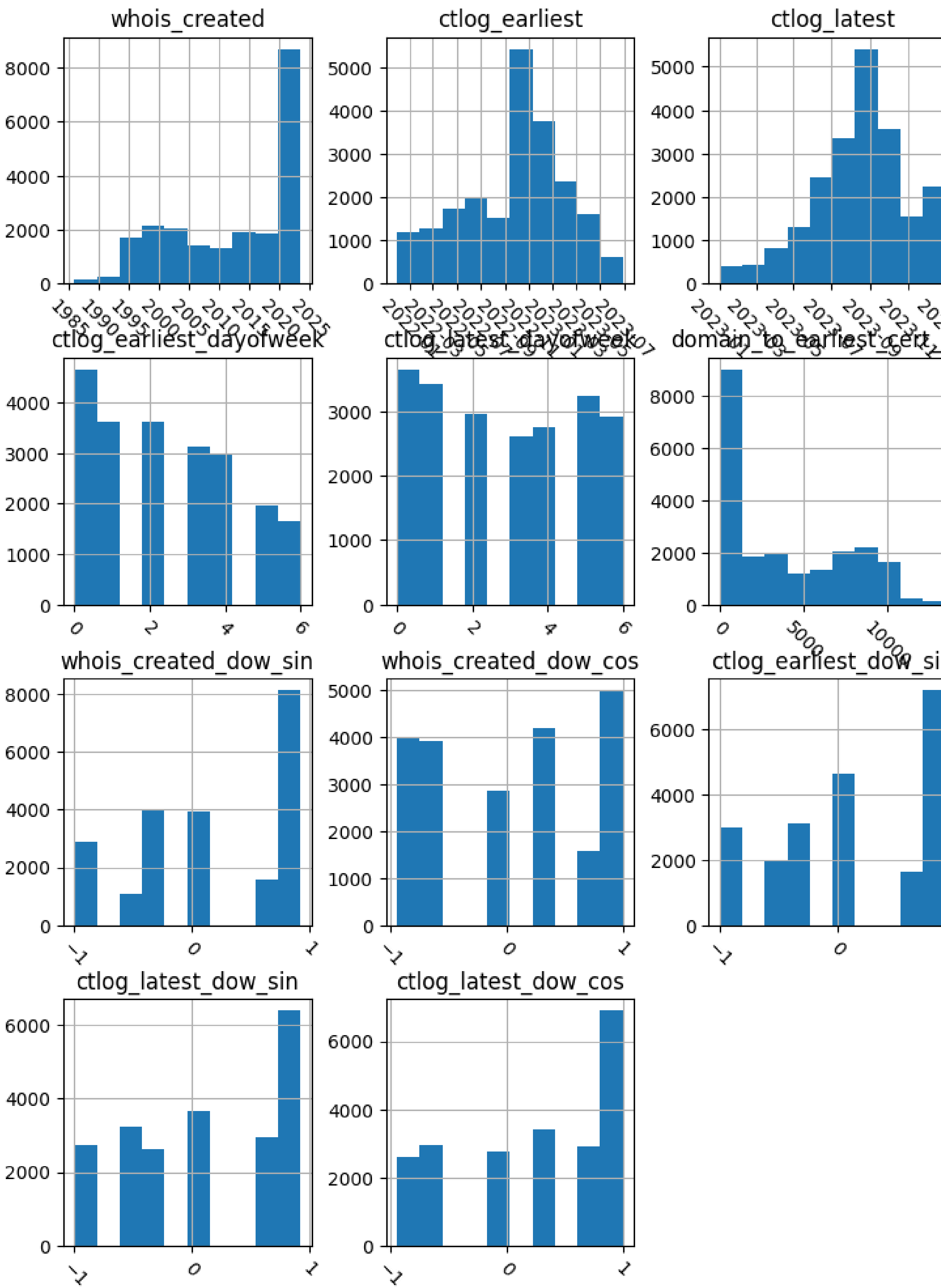

1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
whois_created_dow_cos
−1.0
−0.5
0.0
0.5
1.0
whois_created_dow_sin

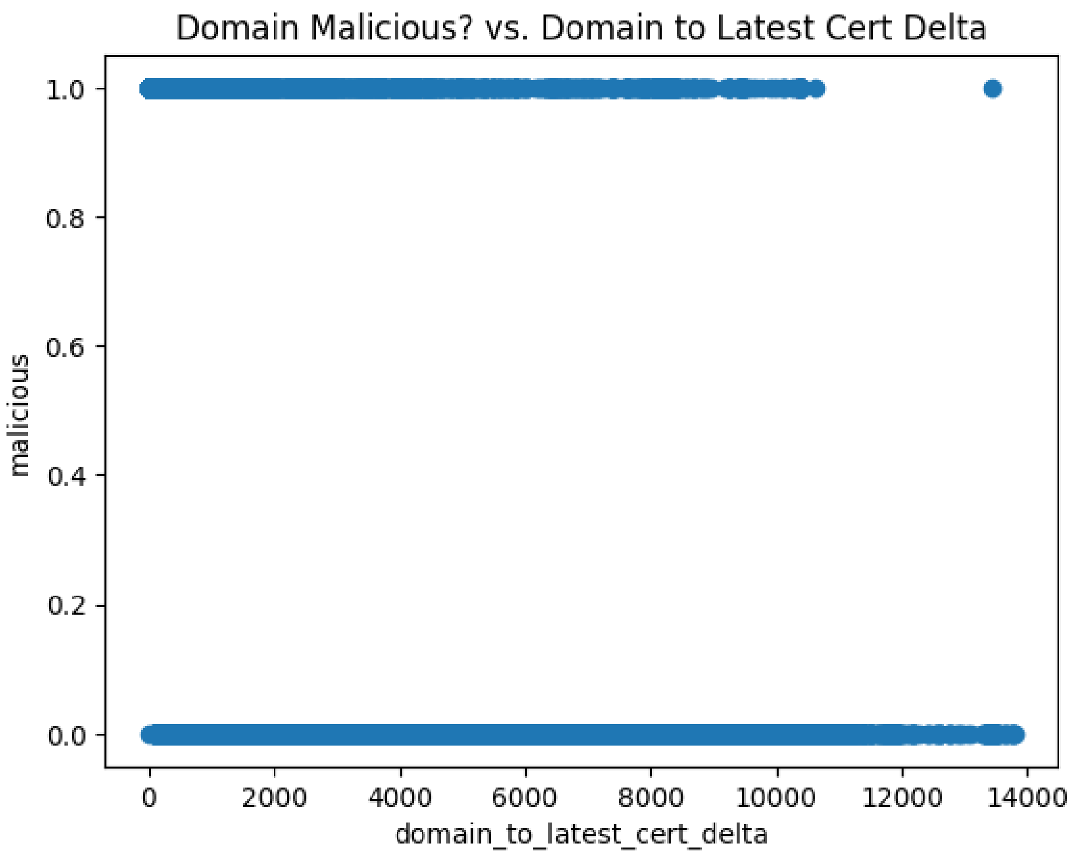


In [4]:

```
# Plot histograms
df.hist(figsize=(12,12), xrot=-45)

# Create a scatter plot of the data, showing malicious and benign domains
# plot the data as a scatter plot
plt.scatter(df["domain_to_earliest_cert_delta"], df["malicious"])
plt.xlabel("domain_to_earliest_cert_delta")
plt.ylabel("malicious")
plt.title("Domain Malicious? vs. Domain to Earliest Cert Delta")
plt.show()
# and for the latest
plt.scatter(df["domain_to_latest_cert_delta"], df["malicious"])
plt.xlabel("domain_to_latest_cert_delta")
plt.ylabel("malicious")
plt.title("Domain Malicious? vs. Domain to Latest Cert Delta")
plt.show()



click.echo(df.head())

# print a count of malicious and benign values
click.echo(df["malicious"].value_counts())
# deduplicate any rows, there shouldn't be any, but just in case
df = df.drop_duplicates()
click.echo(df["malicious"].value_counts())
```

```
click.echo(df.head())

X = df.drop(["malicious","domain", "ctlog_earliest", "ctlog_latest",
"whois_created", "whois_created_dayofweek", "ctlog_earliest_dayofweek"],
axis=1)
X = df.filter(combo_features)
y = df.filter(["malicious"])


click.echo(X.describe())
```


whois_created
ctlog_earliest
ctlog_latest
ctlog_earliest_dayofweek
ctlog_latest_dayofweek
whois_created_dow_sin
whois_created_dow_cos
ctlog_latest_dow_sin
ctlog_latest_dow_cos
0
2000
4000
6000
8000
1000
3000
5000
1985
1990
1995
2000
2005
2010
2015
2020
2025
2
4
6
-1
1




```
                                domain  malicious        whois_created  
0  i-db5p-cor001.api.p001.1drv.com      False 2013-08-05 18:33:50   \
4       soundcloud-pax.pandora.com      False 1993-12-28 05:00:00   
5         joolcomercializadora.com       True 2023-05-22 14:53:50   
6        createpdf-asr.acrobat.com      False 1999-03-16 05:00:00   
8                          popt.in      False 2016-05-14 16:58:55   

       ctlog_earliest        ctlog_latest  ctlog_wildcard  
0 2022-01-24 20:01:58 2023-06-08 20:46:06            True   \
4 2022-05-19 00:00:00 2023-06-19 23:59:59            True   
5 2022-04-06 22:23:24 2023-09-22 23:59:59           False   
6 2022-09-09 00:00:00 2023-10-10 23:59:59            True   
8 2023-01-07 20:36:15 2023-08-15 04:16:52           False   

   whois_created_dayofweek  ctlog_earliest_dayofweek  
ctlog_latest_dayofweek  
0                        0                         0  
3  \
4                        1                         3  
0  
5                        0                         2  
4  
6                        1                         4  
1  
8                        5                         5  
1  
```

```
   domain_to_earliest_cert_delta  domain_to_latest_cert_delta
0                         3095.0                       3595.0  \
4                        10369.0                      10766.0
5                          410.0                        124.0
6                         8578.0                       8975.0
8                         2430.0                       2649.0

   whois_created_dow_sin  whois_created_dow_cos  ctlog_earliest_dow_sin
0               0.000000               1.000000                0.000000  \
4               0.918032               0.396506               -0.340712
5               0.000000               1.000000                0.728010
6               0.918032               0.396506               -0.998199
8              -0.450871               0.892589               -0.450871

   ctlog_earliest_dow_cos  ctlog_latest_dow_sin  ctlog_latest_dow_cos
0                1.000000             -0.340712             -0.940168
4               -0.940168              0.000000              1.000000
5               -0.685567             -0.998199             -0.059997
6               -0.059997              0.918032              0.396506
8                0.892589              0.918032              0.396506
malicious
False    11739
True      9810
Name: count, dtype: int64
malicious
False    11739
True      9810
Name: count, dtype: int64
                            domain  malicious       whois_created
0  i-db5p-cor001.api.p001.1drv.com      False 2013-08-05 18:33:50  \
4       soundcloud-pax.pandora.com      False 1993-12-28 05:00:00
5         joolcomercializadora.com       True 2023-05-22 14:53:50
6        createpdf-asr.acrobat.com      False 1999-03-16 05:00:00
8                          popt.in      False 2016-05-14 16:58:55

       ctlog_earliest        ctlog_latest  ctlog_wildcard
0 2022-01-24 20:01:58 2023-06-08 20:46:06            True  \
4 2022-05-19 00:00:00 2023-06-19 23:59:59            True
5 2022-04-06 22:23:24 2023-09-22 23:59:59           False
6 2022-09-09 00:00:00 2023-10-10 23:59:59            True
8 2023-01-07 20:36:15 2023-08-15 04:16:52           False

   whois_created_dayofweek  ctlog_earliest_dayofweek
ctlog_latest_dayofweek
0                        0                         0
3  \
4                        1                         3
0
```

```
5                        0                          2
4
6                        1                          4
1
8                        5                          5
1

   domain_to_earliest_cert_delta  domain_to_latest_cert_delta
0                         3095.0                       3595.0  \
4                        10369.0                      10766.0
5                          410.0                        124.0
6                         8578.0                       8975.0
8                         2430.0                       2649.0

   whois_created_dow_sin  whois_created_dow_cos  ctlog_earliest_dow_sin
0               0.000000               1.000000                0.000000  \
4               0.918032               0.396506               -0.340712
5               0.000000               1.000000                0.728010
6               0.918032               0.396506               -0.998199
8              -0.450871               0.892589               -0.450871

   ctlog_earliest_dow_cos  ctlog_latest_dow_sin  ctlog_latest_dow_cos
0                1.000000             -0.340712             -0.940168
4               -0.940168              0.000000              1.000000
5               -0.685567             -0.998199             -0.059997
6               -0.059997              0.918032              0.396506
8                0.892589              0.918032              0.396506
       domain_to_earliest_cert_delta
count                   21549.000000
mean                     3742.948397
std                      3694.584062
min                         0.000000
25%                       181.000000
50%                      2637.000000
75%                      7078.000000
max                     13445.000000
```

In [5]:

```
# convert y (malicious) to 1/0 int
y = y.astype('int')
# convert X["ctlog_wildcard"]  to 1/0 int
if "ctlog_wildcard" in X.columns:
    X["ctlog_wildcard"] = X["ctlog_wildcard"].astype('int')

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)


# random forest model
```

```
param_grid = {
    'n_estimators': [50,100,150,200],
    'max_features': ['sqrt', 'log2'],
    'max_depth' : [2,3,4,5],
    'criterion' :['gini', 'entropy']
}
```

In [6]:

```
rf = RandomForestClassifier(random_state=42)
rf_cv = GridSearchCV(estimator=rf, param_grid=param_grid, cv=5, n_jobs=-1)
rf_cv.fit(X_train, y_train.values.ravel())
```

Out[6]:

GridSearchCV

```
GridSearchCV(cv=5, estimator=RandomForestClassifier(random_state=42),
n_jobs=-1,
             param_grid={'criterion': ['gini', 'entropy'],
                         'max_depth': [2, 3, 4, 5],
                         'max_features': ['sqrt', 'log2'],
                         'n_estimators': [50, 100, 150, 200]})
```

estimator: RandomForestClassifier

```
RandomForestClassifier(random_state=42)
```

```
RandomForestClassifier
RandomForestClassifier(random_state=42)
```

In [7]:

```
bp = rf_cv.best_params_
click.echo("Best parameters set found:")
click.echo(bp)
Best parameters set found:
{'criterion': 'gini', 'max_depth': 5, 'max_features': 'sqrt',
'n_estimators': 50}
```

In [8]:

```
rf = RandomForestClassifier(random_state=42,
max_features=bp["max_features"], n_estimators=bp["n_estimators"],
max_depth=bp["max_depth"], criterion=bp["criterion"])
```

In [9]:

```
rf.fit(X_train, y_train.values.ravel())
```

Out[9]:

```
RandomForestClassifier
RandomForestClassifier(max_depth=5, n_estimators=50, random_state=42)
```

In [ ]:

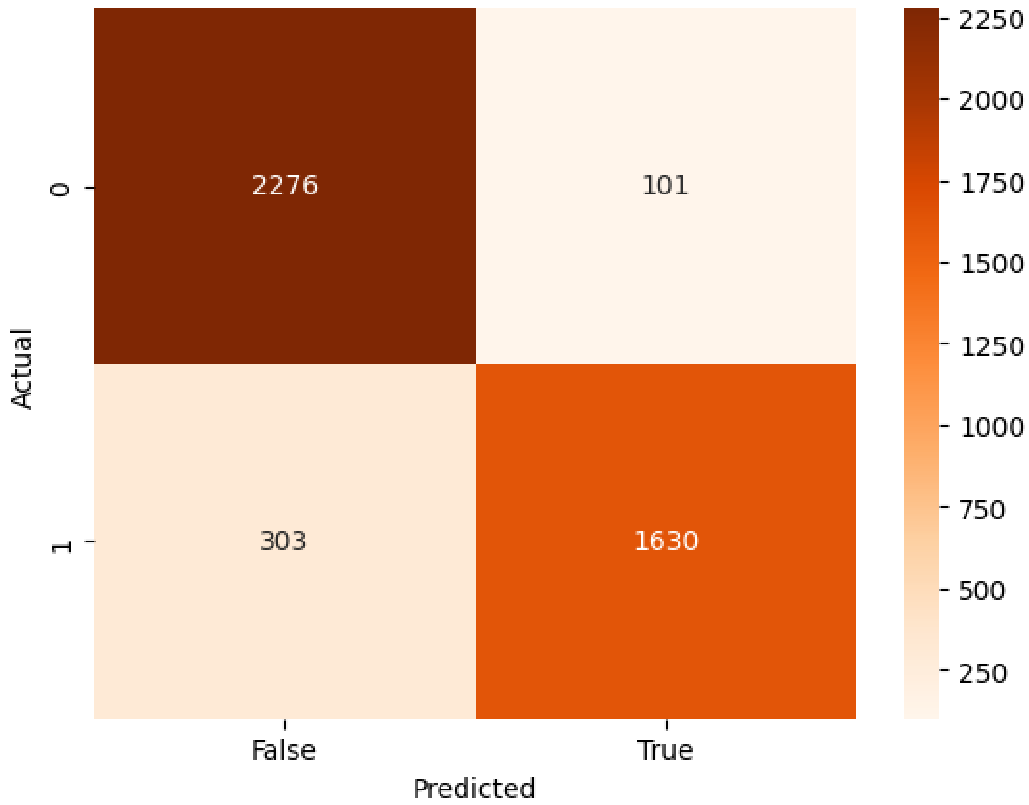
In [10]:

```
# Predict the malicious column using the test data
#add the incepts


y_predicted = rf.predict(X_test)

# Present the results
click.echo("Features selected:")
```

```
click.echo(X.columns)
click.echo("Confusion matrix:")
cm = confusion_matrix(y_test, y_predicted)
click.echo(cm)
click.echo("Classification report:")
click.echo(classification_report(y_test, y_predicted))


# Heatmap of confusion matrix
y_predicted

threshold = 0.5
y_predicted = [y > threshold for y in y_predicted]
data = {'Actual':    y_test.values.flatten(),
        'Predicted': y_predicted
        }

# Generate a confusion matrix and heatmap to evaluate the Type I and Type
II errors/ FP/FN etc.
df = pd.DataFrame(data, columns=['Actual','Predicted'])
cm2 = pd.crosstab(df['Actual'], df['Predicted'], rownames=['Actual'],
colnames=['Predicted'])
fig = sns.heatmap(cm2, annot=True, cmap='Oranges', fmt='g')
fig
Features selected:
Index(['domain_to_earliest_cert_delta'], dtype='object')
Confusion matrix:
[[2276  101]
 [ 303 1630]]
Classification report:
              precision    recall  f1-score   support

           0       0.88      0.96      0.92      2377
           1       0.94      0.84      0.89      1933

    accuracy                           0.91      4310
   macro avg       0.91      0.90      0.90      4310
weighted avg       0.91      0.91      0.91      4310
```

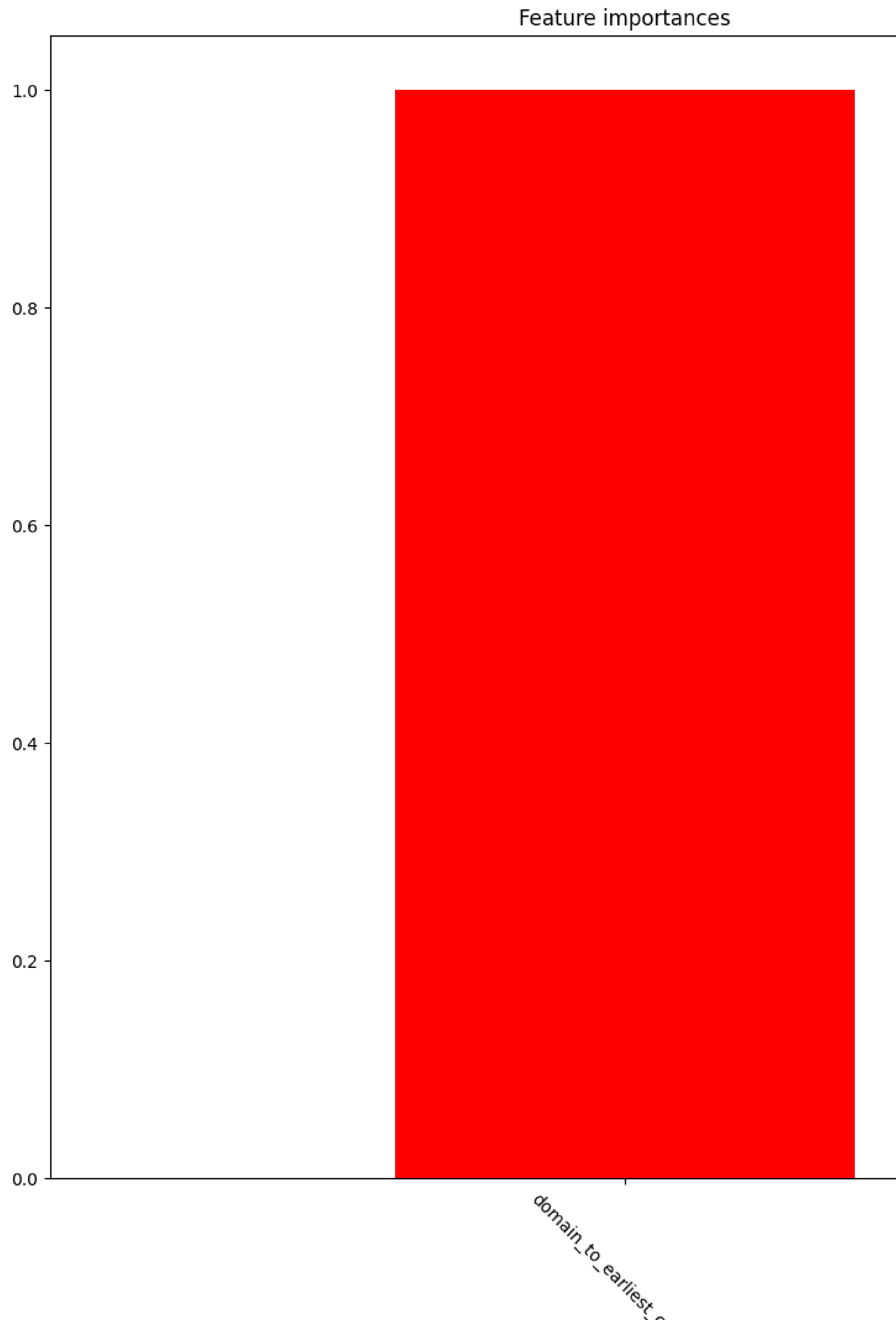
Out[10]:

```
<Axes: xlabel='Predicted', ylabel='Actual'>
```



In [11]:

```
# plot the feature importances
importances = rf.feature_importances_
std = np.std([tree.feature_importances_ for tree in rf.estimators_],
axis=0)

indices = np.argsort(importances)[::-1]
# Print the feature ranking
click.echo("Feature ranking:")
for f in range(X.shape[1]):
    click.echo("%d. feature %s (%f)" % (f + 1, combo_features[indices[f]],
importances[indices[f]]))

# Plot the feature importances of the forest
plt.figure(figsize=(12,12))
plt.title("Feature importances")
plt.bar(range(X.shape[1]), importances[indices], color="r",
yerr=std[indices], align="center")
plt.xticks(range(X.shape[1]), X.columns[indices], rotation=-45)
plt.xlim([-1, X.shape[1]])
plt.show()
Feature ranking:
1. feature domain_to_earliest_cert_delta (1.000000)
```


Feature importances
1.0
0.8
0.6
0.4
0.2
0.0
domain_to_earliest_

II. Feature Set A

In [1]:

```
import click
import pandas as pd
import glob
import os
#import sklearn
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.metrics import accuracy_score, f1_score, confusion_matrix,
classification_report, roc_auc_score, roc_curve, auc
from sklearn.preprocessing import StandardScaler
from scipy import stats
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
import statsmodels.api as sm
import matplotlib.pyplot as plt
from datetime import datetime, date
# qqplot of the data
import seaborn as sns
import math
import numpy as np
import utils

combo_features = ['domain_to_earliest_cert_delta',
'ctlog_earliest_dow_sin', 'ctlog_earliest_dow_cos']

path = "../data/"
filename = None

epoch = datetime(1970,1,1)
# open the training data file, from ../data/ for the most recent merged
training data file saved by prepare-training-data-parallel.py
full_path = path + "merged_20230705-104357_training_adorned-engineered.csv"

# get latest file matching the glob
if not filename:
    filename = max(glob.glob(full_path), key=os.path.getctime)
    click.echo(f"Using {filename} as training data")
    df = pd.read_csv(filename, sep=",")

# randomize the rows
df = df.sample(frac=1, random_state=42).reset_index(drop=True)

click.echo(df.shape)
click.echo(df.columns)

""" # From prepare-training-data-parallel.py:
# Columns:
url
```

```
verification_time
domain
malicious
dateadded
whois_created
soa_timestamp
ctlog_earliest
ctlog_latest
ctlog_wildcard
whois_created_dayofweek,
ctlog_earliest_dayofweek,
domain_to_cert_delta
"""


# Data cleaning - there may be some epoch values in the whois_created
#  & ct_log_earliest columns, so we need to remove those rows

# convert the whois_created and ctlog_earliest, ctlog_latest columns to
datetime

df = df.loc[df["whois_created"] != ""]
df = df.loc[df["ctlog_earliest"] != ""]
df = df.loc[df["ctlog_latest"] != ""]

# some dates are have nanoseconds in them, so we need to remove the
nanosecond part
df["whois_created"] = df["whois_created"].str.split(".", n = 1, expand =
True)[0]
# some dates has additional timezone information, so we need to remove that
df["whois_created"] = df["whois_created"].apply(lambda x: " ".join(
str(x).split(" ")[:2]))


# convert the whois_created and ct_log_earliest columns to datetime
df = df.astype({"whois_created": 'datetime64[ns]', "ctlog_earliest":
'datetime64[ns]', "ctlog_latest": 'datetime64[ns]'})
df = df.loc[df["whois_created"].ne(pd.Timestamp('1970-01-01 00:00:00'))]
df = df.loc[df["ctlog_earliest"].ne(pd.Timestamp('1970-01-01 00:00:00'))]
df = df.loc[df["ctlog_latest"].ne(pd.Timestamp('1970-01-01 00:00:00'))]


# malicious is a bool
df['malicious'] = df['malicious'].astype('bool')
df['domain'] = df['domain'].astype('string')
```

```
Using ../data/merged_20230705-104357_training_adorned-engineered.csv as
training data
(35438, 13)
Index(['url', 'verification_time', 'domain', 'malicious', 'dateadded',
```

```
       'whois_created', 'soa_timestamp', 'ctlog_earliest', 'ctlog_latest',
       'ctlog_wildcard', 'whois_created_dayofweek',
'ctlog_earliest_dayofweek',
       'domain_to_cert_delta'],
      dtype='object')
```

In [2]:

```
#########
# Feature engineering
#########

# remove any rows where the whois_created or ctlog_earliest columns are
null
df = df.loc[df["whois_created"].notnull()]
df = df.loc[df["ctlog_earliest"].notnull()]

# if the data is missing the "whois_created_dayofweek" or
"ctlog_earliest_dayofweek" columns, then add them
# whois created day of the week 0 = Monday, 6 = Sunday
df["whois_created_dayofweek"] = df["whois_created"].apply(
        lambda x: x.weekday() if isinstance(x, date) else None
    )

# cert valid day of the week 0 = Monday, 6 = Sunday
df["ctlog_earliest_dayofweek"] = df["ctlog_earliest"].apply(
        lambda x: x.weekday() if isinstance(x, date) else None
    )

df["ctlog_latest_dayofweek"] = df["ctlog_latest"].apply(
        lambda x: x.weekday() if isinstance(x, date) else None
    )

# set data type of the day of week columns to int
df["whois_created_dayofweek"] =
df["whois_created_dayofweek"].astype("int64")
df["ctlog_earliest_dayofweek"] =
df["ctlog_earliest_dayofweek"].astype("int64")
df["ctlog_latest_dayofweek"] = df["ctlog_latest_dayofweek"].astype("int64")

# domain to cert delta
df["domain_to_earliest_cert_delta"] = df.apply(
        lambda x: utils.get_days_delta(
            x["ctlog_earliest"],
            x["whois_created"]
            if x["whois_created"] and x["ctlog_earliest"]
            else None,
        ),
        axis=1,
    )
```

```
# set the data type of the domain_to_cert_delta column to float
df["domain_to_earliest_cert_delta"] =
df["domain_to_earliest_cert_delta"].astype("float64")

# domain to latest cert delta
df["domain_to_latest_cert_delta"] = df.apply(
        lambda x: utils.get_days_delta(
            x["ctlog_latest"],
            x["whois_created"]
            if x["whois_created"] and x["ctlog_latest"]
            else None,
        ),
        axis=1,
    )

# set the data type of the domain_to_cert_delta column to float
df["domain_to_latest_cert_delta"] =
df["domain_to_latest_cert_delta"].astype("float64")

# drop columns we imported that we will never use
df = df.drop(columns=["url", "verification_time", "dateadded",
"soa_timestamp","domain_to_cert_delta"])

click.echo(df.head())
click.echo(df.describe(include='all'))
click.echo(df.dtypes)
                            domain  malicious       whois_created
0  i-db5p-cor001.api.p001.1drv.com      False 2013-08-05 18:33:50  \
4       soundcloud-pax.pandora.com      False 1993-12-28 05:00:00
5         joolcomercializadora.com       True 2023-05-22 14:53:50
6        createpdf-asr.acrobat.com      False 1999-03-16 05:00:00
8                          popt.in      False 2016-05-14 16:58:55

       ctlog_earliest        ctlog_latest  ctlog_wildcard
0 2022-01-24 20:01:58 2023-06-08 20:46:06            True  \
4 2022-05-19 00:00:00 2023-06-19 23:59:59            True
5 2022-04-06 22:23:24 2023-09-22 23:59:59           False
6 2022-09-09 00:00:00 2023-10-10 23:59:59            True
8 2023-01-07 20:36:15 2023-08-15 04:16:52           False

   whois_created_dayofweek  ctlog_earliest_dayofweek
ctlog_latest_dayofweek
0                        0                          0
3  \
4                        1                          3
0
5                        0                          2
4
```

```
6                        1                         4
1
8                        5                         5
1

   domain_to_earliest_cert_delta  domain_to_latest_cert_delta
0                        -3095.0                      -3595.0
4                       -10369.0                     -10766.0
5                          410.0                       -124.0
6                        -8578.0                      -8975.0
8                        -2430.0                      -2649.0
                domain malicious                  whois_created
count            21549     21549                          21549  \
unique           21536         2                            NaN
top     www.mediafire.com     False                         NaN
freq                 2     11739                            NaN
mean               NaN       NaN  2012-10-03 12:56:32.335050496
min                NaN       NaN            1986-01-09 00:00:00
25%                NaN       NaN            2003-05-25 13:35:05
50%                NaN       NaN            2015-05-07 23:56:05
75%                NaN       NaN            2023-03-20 15:03:16
max                NaN       NaN            2023-07-03 08:21:24
std                NaN       NaN                            NaN

                         ctlog_earliest                    ctlog_latest
count                             21549                           21549  \
unique                              NaN                             NaN
top                                 NaN                             NaN
freq                                NaN                             NaN
mean    2022-09-26 15:45:50.943570432  2023-08-14 17:49:06.400900352
min               2021-11-30 05:24:28             2023-01-01 18:42:11
25%               2022-06-24 13:47:12             2023-07-02 08:11:07
50%               2022-10-18 21:00:14             2023-08-21 21:40:11
75%               2022-12-14 00:00:00             2023-09-21 19:41:38
max               2023-06-28 04:36:22             2023-12-31 23:59:59
std                                 NaN                             NaN

       ctlog_wildcard  whois_created_dayofweek  ctlog_earliest_dayofweek
count           21549             21549.000000              21549.000000  \
unique              2                      NaN                       NaN
top             False                      NaN                       NaN
freq            13032                      NaN                       NaN
mean              NaN                 2.332823                  2.399462
min               NaN                 0.000000                  0.000000
25%               NaN                 1.000000                  1.000000
50%               NaN                 2.000000                  2.000000
75%               NaN                 4.000000                  4.000000
max               NaN                 6.000000                  6.000000
std               NaN                 1.775043                  1.897252
```

```
        ctlog_latest_dayofweek  domain_to_earliest_cert_delta
count             21549.000000                   21549.000000  \
unique                     NaN                            NaN
top                        NaN                            NaN
freq                       NaN                            NaN
mean                  2.873080                   -3645.602070
min                   0.000000                  -13445.000000
25%                   1.000000                   -7078.000000
50%                   3.000000                   -2637.000000
75%                   5.000000                      69.000000
max                   6.000000                     524.000000
std                   2.057394                    3790.677119

        domain_to_latest_cert_delta
count                  21549.000000
unique                          NaN
top                             NaN
freq                            NaN
mean                   -3967.678222
min                   -13798.000000
25%                    -7421.000000
50%                    -3009.000000
75%                     -144.000000
max                      135.000000
std                     3852.703681
domain                               string[python]
malicious                                      bool
whois_created                        datetime64[ns]
ctlog_earliest                       datetime64[ns]
ctlog_latest                         datetime64[ns]
ctlog_wildcard                                 bool
whois_created_dayofweek                       int64
ctlog_earliest_dayofweek                      int64
ctlog_latest_dayofweek                        int64
domain_to_earliest_cert_delta               float64
domain_to_latest_cert_delta                 float64
dtype: object
```

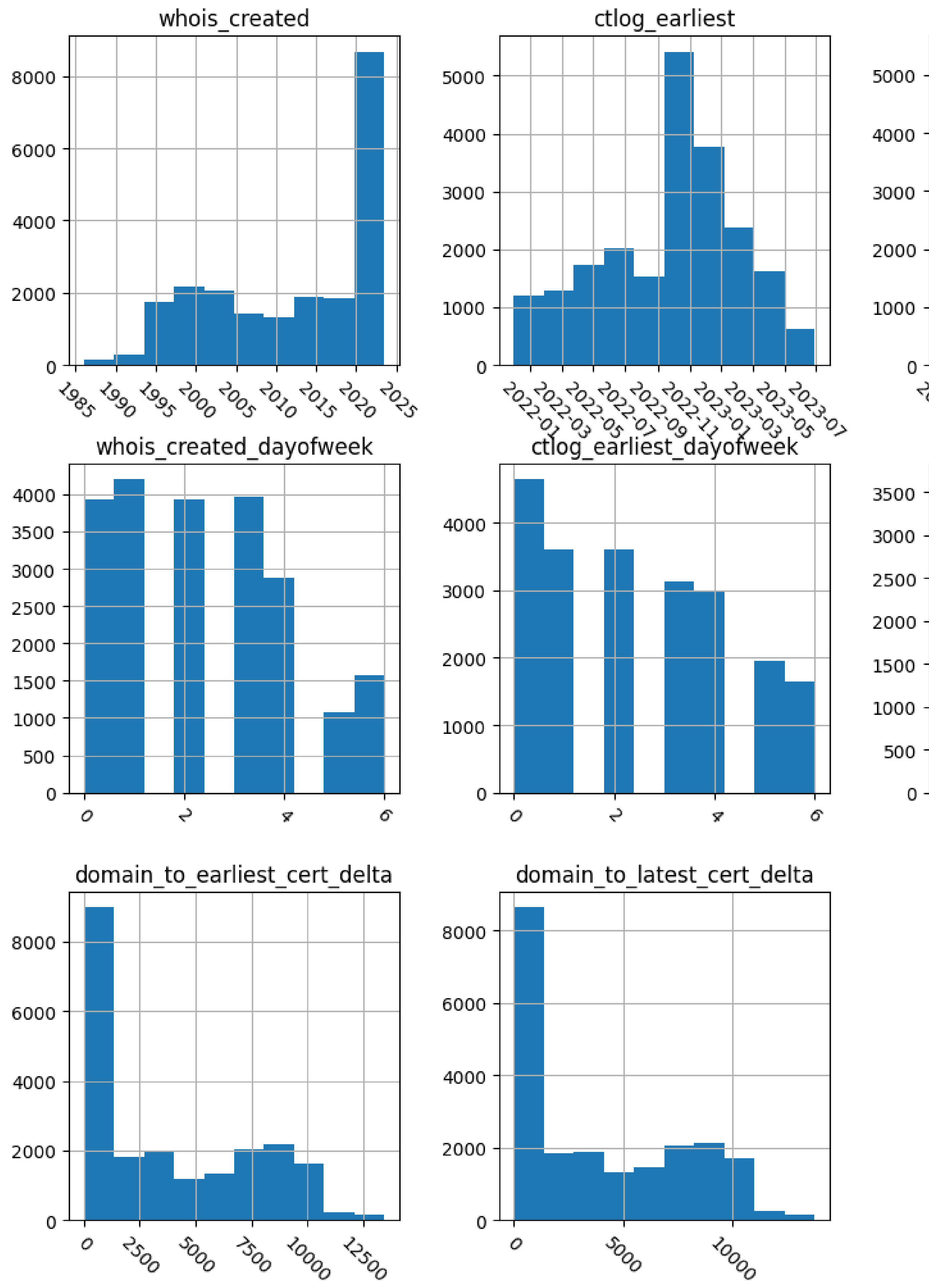
In [3]:

```
# absolute value of the domain_to_cert_delta
df["domain_to_earliest_cert_delta"] =
abs(df["domain_to_earliest_cert_delta"])
df["domain_to_latest_cert_delta"] = abs(df["domain_to_latest_cert_delta"])

df.hist(figsize=(12,12), xrot=-45)

col_index = df.columns.get_loc("whois_created_dayofweek")
df["whois_created_dow_sin"] = df.apply(lambda row:
math.sin(360/7*(row[col_index])),axis=1)
```

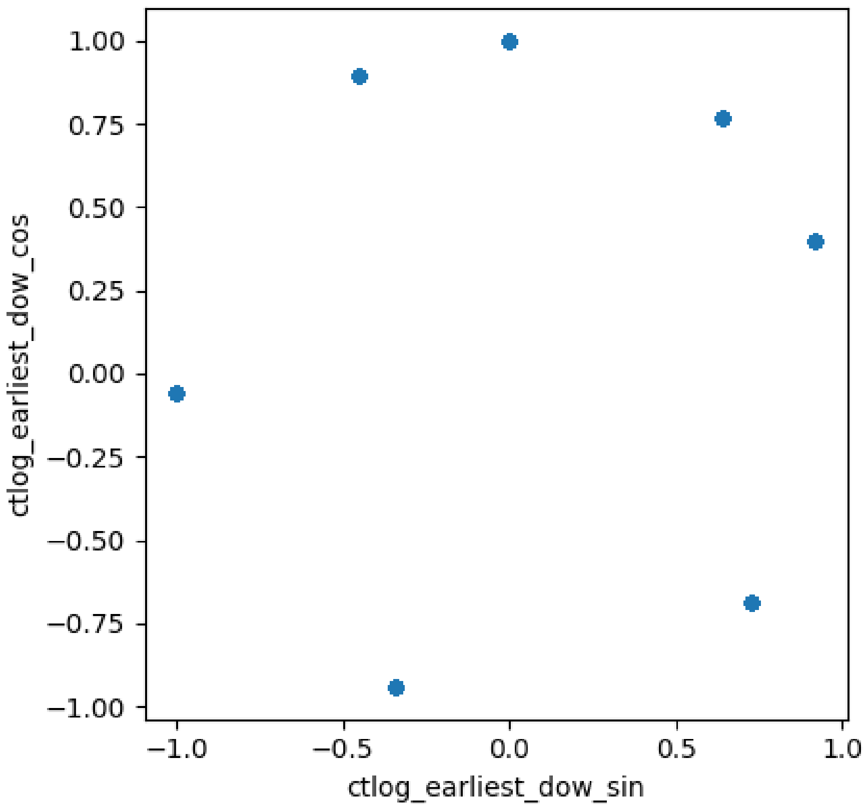
```
df["whois_created_dow_cos"] = df.apply(lambda row:
math.cos(360/7*(row[col_index])),axis=1)

col_index = df.columns.get_loc("ctlog_earliest_dayofweek")
df["ctlog_earliest_dow_sin"] = df.apply(lambda row:
math.sin(360/7*(row[col_index])),axis=1)
df["ctlog_earliest_dow_cos"] = df.apply(lambda row:
math.cos(360/7*(row[col_index])),axis=1)

col_index = df.columns.get_loc("ctlog_latest_dayofweek")
df["ctlog_latest_dow_sin"] = df.apply(lambda row:
math.sin(360/7*(row[col_index])),axis=1)
df["ctlog_latest_dow_cos"] = df.apply(lambda row:
math.cos(360/7*(row[col_index])),axis=1)


df.plot.scatter(x="ctlog_earliest_dow_sin",
y="ctlog_earliest_dow_cos").set_aspect('equal')
df.plot.scatter(x="whois_created_dow_sin",
y="whois_created_dow_cos").set_aspect('equal')
df.plot.scatter(x="ctlog_latest_dow_sin",
y="ctlog_latest_dow_cos").set_aspect('equal')

"""
# add one hot encode ctlog_earliest_not_before_dayofweek
df = pd.get_dummies(
        df,
        columns=["ctlog_earliest_dayofweek"],
        prefix=["ctlog_earliest_dow"],
    )
# add one hot encode whois_created_dayofweek to columns
df = pd.get_dummies(
        df, columns=["whois_created_dayofweek"], prefix="whois_dow"
    )
    """

# Summary statistics
click.echo(df.describe(include='all'))
                   domain malicious                   whois_created
count               21549     21549                           21549  \
unique              21536         2                             NaN
top     www.mediafire.com     False                             NaN
freq                    2     11739                             NaN
mean                  NaN       NaN  2012-10-03 12:56:32.335050496
min                   NaN       NaN            1986-01-09 00:00:00
25%                   NaN       NaN            2003-05-25 13:35:05
50%                   NaN       NaN            2015-05-07 23:56:05
75%                   NaN       NaN            2023-03-20 15:03:16
max                   NaN       NaN            2023-07-03 08:21:24
```

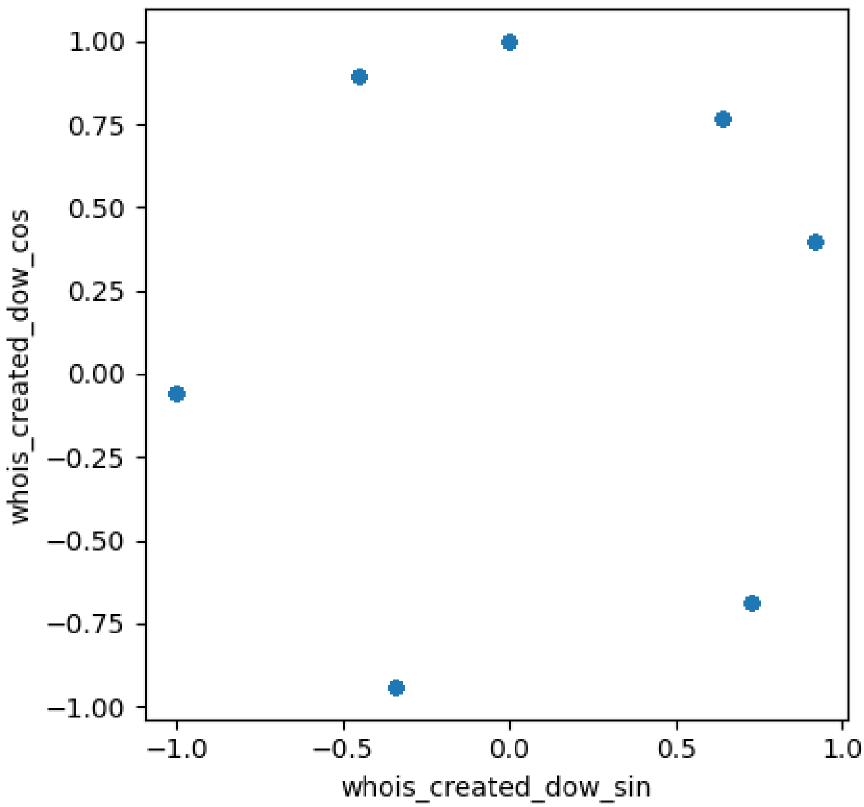
```
std                  NaN       NaN                             NaN

                       ctlog_earliest                  ctlog_latest
count                           21549                         21549  \
unique                            NaN                           NaN
top                               NaN                           NaN
freq                              NaN                           NaN
mean    2022-09-26 15:45:50.943570432 2023-08-14 17:49:06.400900352
min               2021-11-30 05:24:28           2023-01-01 18:42:11
25%               2022-06-24 13:47:12           2023-07-02 08:11:07
50%               2022-10-18 21:00:14           2023-08-21 21:40:11
75%               2022-12-14 00:00:00           2023-09-21 19:41:38
max               2023-06-28 04:36:22           2023-12-31 23:59:59
std                               NaN                           NaN

       ctlog_wildcard  whois_created_dayofweek  ctlog_earliest_dayofweek
count           21549             21549.000000              21549.000000  \
unique              2                      NaN                       NaN
top             False                      NaN                       NaN
freq            13032                      NaN                       NaN
mean              NaN                 2.332823                  2.399462
min               NaN                 0.000000                  0.000000
25%               NaN                 1.000000                  1.000000
50%               NaN                 2.000000                  2.000000
75%               NaN                 4.000000                  4.000000
max               NaN                 6.000000                  6.000000
std               NaN                 1.775043                  1.897252

        ctlog_latest_dayofweek  domain_to_earliest_cert_delta
count             21549.000000                   21549.000000  \
unique                     NaN                            NaN
top                        NaN                            NaN
freq                       NaN                            NaN
mean                  2.873080                    3742.948397
min                   0.000000                       0.000000
25%                   1.000000                     181.000000
50%                   3.000000                    2637.000000
75%                   5.000000                    7078.000000
max                   6.000000                   13445.000000
std                   2.057394                    3694.584062

        domain_to_latest_cert_delta  whois_created_dow_sin
count                  21549.000000           21549.000000  \
unique                          NaN                    NaN
top                             NaN                    NaN
freq                            NaN                    NaN
mean                    3969.491206               0.140419
min                        0.000000              -0.998199
25%                      144.000000              -0.340712
```

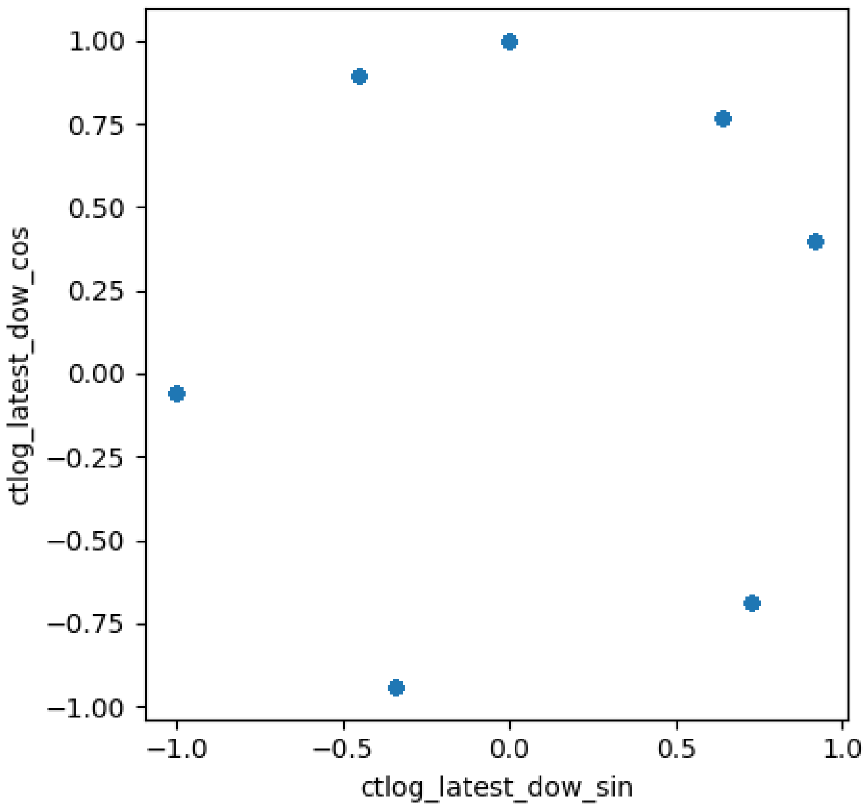
```
50%                   3009.000000              0.000000
75%                   7421.000000              0.728010
max                  13798.000000              0.918032
std                   3850.835626              0.659922

        whois_created_dow_cos  ctlog_earliest_dow_sin
ctlog_earliest_dow_cos
count            21549.000000            21549.000000
21549.000000  \
unique                    NaN                     NaN
NaN
top                       NaN                     NaN
NaN
freq                      NaN                     NaN
NaN
mean                 0.054288                0.095357
0.161451
min                 -0.940168               -0.998199               -
0.940168
25%                 -0.685567               -0.340712               -
0.685567
50%                  0.396506                0.000000
0.396506
75%                  0.767830                0.728010
0.892589
max                  1.000000                0.918032
1.000000
std                  0.736128                0.651782
0.734891

        ctlog_latest_dow_sin  ctlog_latest_dow_cos
count           21549.000000          21549.000000
unique                   NaN                   NaN
top                      NaN                   NaN
freq                     NaN                   NaN
mean                0.096253              0.255578
min                -0.998199             -0.940168
25%                -0.450871             -0.685567
50%                 0.000000              0.396506
75%                 0.728010              0.892589
max                 0.918032              1.000000
std                 0.651597              0.707728
```



whois_created
0
2000
4000
6000
8000
1985
1990
1995
2000
2005
2010
2015
2020
2025
ctlog_earliest
0
1000
2000
3000
4000
5000
2022-01
2022-03
2022-05
2022-07
2022-09
2022-11
2023-01
2023-03
2023-05
2023-07
whois_created_dayofweek
0
500
1000
1500
2000
2500
3000
3500
4000
0
2
4
6
ctlog_earliest_dayofweek
0
1000
2000
3000
4000
0
2
4
6
domain_to_earliest_cert_delta
0
2000
4000
6000
8000
0
2500
5000
7500
10000
12500
domain_to_latest_cert_delta
0
2000
4000
6000
8000
0
5000
10000


1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
ctlog_earliest_dow_cos
−1.0
−0.5
0.0
0.5
1.0
ctlog_earliest_dow_sin

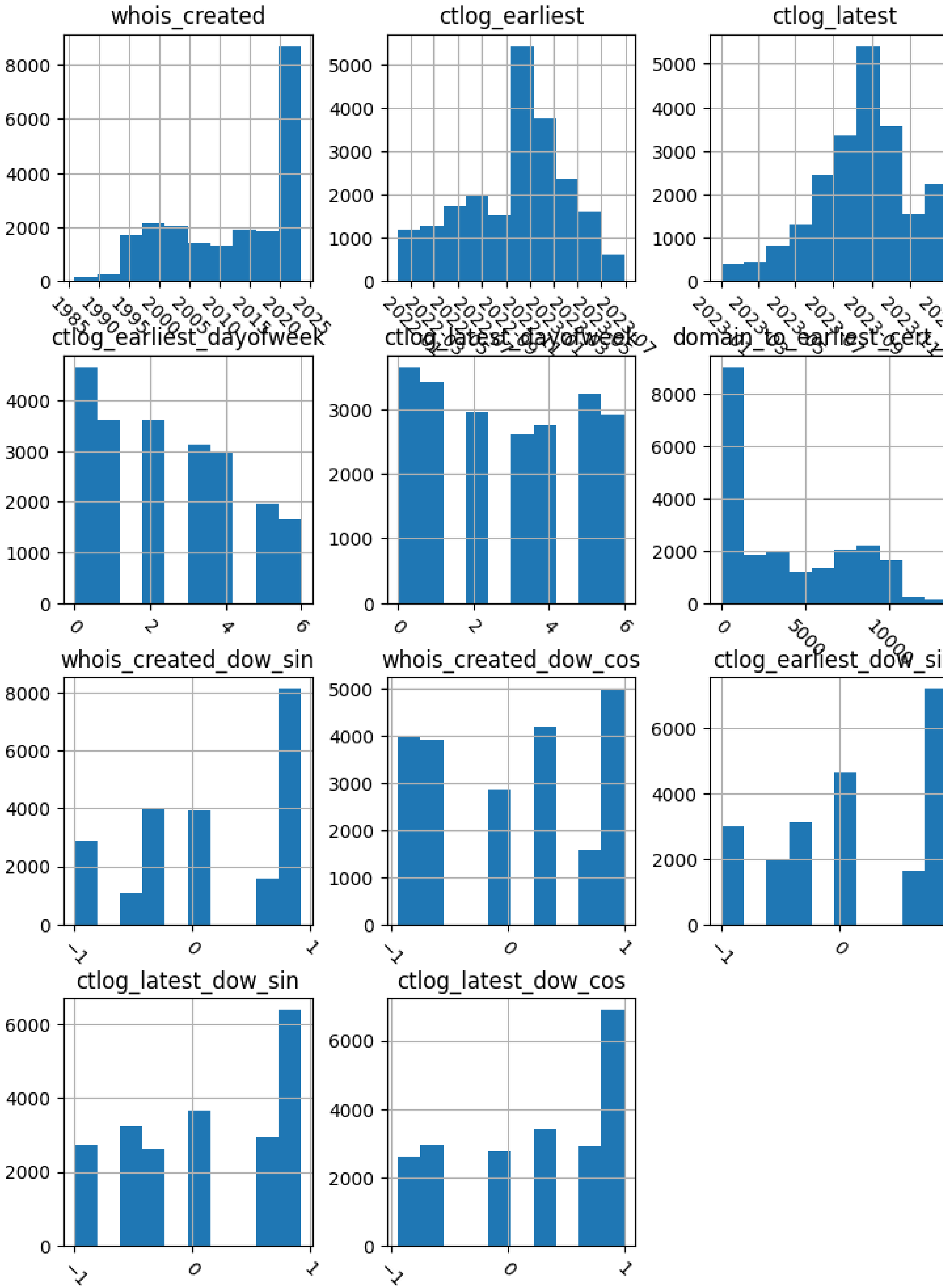

1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
whois_created_dow_cos
−1.0
−0.5
0.0
0.5
1.0
whois_created_dow_sin

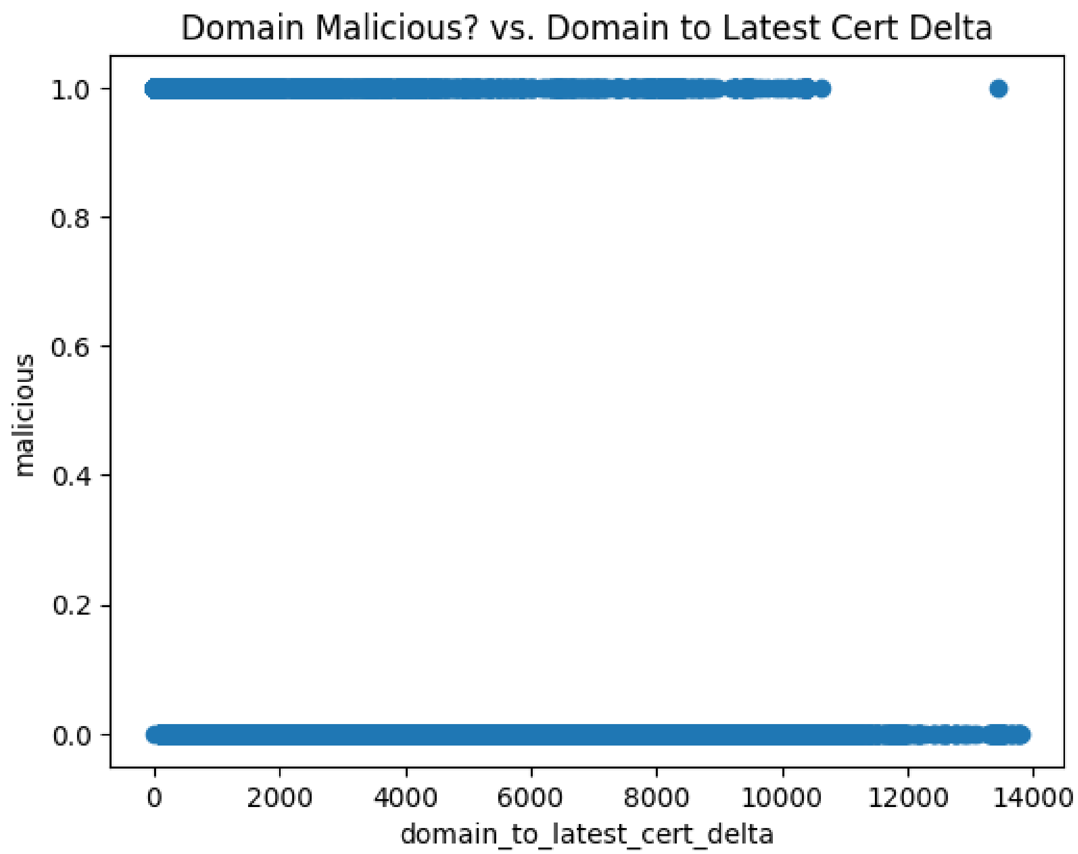


In [4]:

```
# Plot histograms
df.hist(figsize=(12,12), xrot=-45)

# Create a scatter plot of the data, showing malicious and benign domains
# plot the data as a scatter plot
plt.scatter(df["domain_to_earliest_cert_delta"], df["malicious"])
plt.xlabel("domain_to_earliest_cert_delta")
plt.ylabel("malicious")
plt.title("Domain Malicious? vs. Domain to Earliest Cert Delta")
plt.show()
# and for the latest
plt.scatter(df["domain_to_latest_cert_delta"], df["malicious"])
plt.xlabel("domain_to_latest_cert_delta")
plt.ylabel("malicious")
plt.title("Domain Malicious? vs. Domain to Latest Cert Delta")
plt.show()



click.echo(df.head())

# print a count of malicious and benign values
click.echo(df["malicious"].value_counts())
# deduplicate any rows, there shouldn't be any, but just in case
df = df.drop_duplicates()
click.echo(df["malicious"].value_counts())
```

```
click.echo(df.head())

X = df.drop(["malicious","domain", "ctlog_earliest", "ctlog_latest",
"whois_created", "whois_created_dayofweek", "ctlog_earliest_dayofweek"],
axis=1)
X = df.filter(combo_features)
y = df.filter(["malicious"])


click.echo(X.describe())
```


whois_created
ctlog_earliest
ctlog_latest
ctlog_earliest_dayofweek
ctlog_latest_dayofweek
domain_to_earliest_cert_
whois_created_dow_sin
whois_created_dow_cos
ctlog_earliest_dow_si
ctlog_latest_dow_sin
ctlog_latest_dow_cos
0
2000
4000
6000
8000
1000
3000
5000
1985
1990
1995
2000
2005
2010
2015
2020
2025
2
4
6
5000
10000
-1
1




```
                              domain  malicious        whois_created
0  i-db5p-cor001.api.p001.1drv.com       False 2013-08-05 18:33:50   \
4       soundcloud-pax.pandora.com       False 1993-12-28 05:00:00
5         joolcomercializadora.com        True 2023-05-22 14:53:50
6        createpdf-asr.acrobat.com       False 1999-03-16 05:00:00
8                          popt.in       False 2016-05-14 16:58:55

        ctlog_earliest         ctlog_latest  ctlog_wildcard  
0 2022-01-24 20:01:58 2023-06-08 20:46:06            True   \
4 2022-05-19 00:00:00 2023-06-19 23:59:59            True
5 2022-04-06 22:23:24 2023-09-22 23:59:59           False
6 2022-09-09 00:00:00 2023-10-10 23:59:59            True
8 2023-01-07 20:36:15 2023-08-15 04:16:52           False

   whois_created_dayofweek  ctlog_earliest_dayofweek  
ctlog_latest_dayofweek
0                        0                          0   
3  \
4                        1                          3   
0
5                        0                          2   
4
6                        1                          4   
1
8                        5                          5   
1
```

```
   domain_to_earliest_cert_delta  domain_to_latest_cert_delta
0                         3095.0                       3595.0  \
4                        10369.0                      10766.0
5                          410.0                        124.0
6                         8578.0                       8975.0
8                         2430.0                       2649.0

   whois_created_dow_sin  whois_created_dow_cos  ctlog_earliest_dow_sin
0               0.000000               1.000000                0.000000  \
4               0.918032               0.396506               -0.340712
5               0.000000               1.000000                0.728010
6               0.918032               0.396506               -0.998199
8              -0.450871               0.892589               -0.450871

   ctlog_earliest_dow_cos  ctlog_latest_dow_sin  ctlog_latest_dow_cos
0                1.000000             -0.340712             -0.940168
4               -0.940168              0.000000              1.000000
5               -0.685567             -0.998199             -0.059997
6               -0.059997              0.918032              0.396506
8                0.892589              0.918032              0.396506
malicious
False    11739
True      9810
Name: count, dtype: int64
malicious
False    11739
True      9810
Name: count, dtype: int64
                            domain  malicious       whois_created
0  i-db5p-cor001.api.p001.1drv.com      False 2013-08-05 18:33:50  \
4       soundcloud-pax.pandora.com      False 1993-12-28 05:00:00
5         joolcomercializadora.com       True 2023-05-22 14:53:50
6        createpdf-asr.acrobat.com      False 1999-03-16 05:00:00
8                          popt.in      False 2016-05-14 16:58:55

       ctlog_earliest        ctlog_latest  ctlog_wildcard
0 2022-01-24 20:01:58 2023-06-08 20:46:06            True  \
4 2022-05-19 00:00:00 2023-06-19 23:59:59            True
5 2022-04-06 22:23:24 2023-09-22 23:59:59           False
6 2022-09-09 00:00:00 2023-10-10 23:59:59            True
8 2023-01-07 20:36:15 2023-08-15 04:16:52           False

   whois_created_dayofweek  ctlog_earliest_dayofweek
ctlog_latest_dayofweek
0                        0                         0
3  \
4                        1                         3
0
```

```
5                    0                          2
4
6                    1                          4
1
8                    5                          5
1

   domain_to_earliest_cert_delta  domain_to_latest_cert_delta
0                         3095.0                       3595.0  \
4                        10369.0                      10766.0
5                          410.0                        124.0
6                         8578.0                       8975.0
8                         2430.0                       2649.0

   whois_created_dow_sin  whois_created_dow_cos  ctlog_earliest_dow_sin
0               0.000000               1.000000                0.000000  \
4               0.918032               0.396506               -0.340712
5               0.000000               1.000000                0.728010
6               0.918032               0.396506               -0.998199
8              -0.450871               0.892589               -0.450871

   ctlog_earliest_dow_cos  ctlog_latest_dow_sin  ctlog_latest_dow_cos
0                1.000000             -0.340712             -0.940168
4               -0.940168              0.000000              1.000000
5               -0.685567             -0.998199             -0.059997
6               -0.059997              0.918032              0.396506
8                0.892589              0.918032              0.396506
       domain_to_earliest_cert_delta  ctlog_earliest_dow_sin
count                   21549.000000            21549.000000  \
mean                     3742.948397                0.095357
std                      3694.584062                0.651782
min                         0.000000               -0.998199
25%                       181.000000               -0.340712
50%                      2637.000000                0.000000
75%                      7078.000000                0.728010
max                     13445.000000                0.918032

       ctlog_earliest_dow_cos
count            21549.000000
mean                 0.161451
std                  0.734891
min                 -0.940168
25%                 -0.685567
50%                  0.396506
75%                  0.892589
max                  1.000000
```

In [5]:

```
# convert y (malicious) to 1/0 int
y = y.astype('int')
```

```
# convert X["ctlog_wildcard"]  to 1/0 int
if "ctlog_wildcard" in X.columns:
    X["ctlog_wildcard"] = X["ctlog_wildcard"].astype('int')

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)


# random forest model

param_grid = {
    'n_estimators': [50,100,150,200],
    'max_features': ['sqrt', 'log2'],
    'max_depth' : [2,3,4,5],
    'criterion' :['gini', 'entropy']
}
```

In [6]:

```
rf = RandomForestClassifier(random_state=42)
rf_cv = GridSearchCV(estimator=rf, param_grid=param_grid, cv=5, n_jobs=-1)
rf_cv.fit(X_train, y_train.values.ravel())
```

Out[6]:

GridSearchCV

```
GridSearchCV(cv=5, estimator=RandomForestClassifier(random_state=42),
n_jobs=-1,
             param_grid={'criterion': ['gini', 'entropy'],
                         'max_depth': [2, 3, 4, 5],
                         'max_features': ['sqrt', 'log2'],
                         'n_estimators': [50, 100, 150, 200]})
```

estimator: RandomForestClassifier

```
RandomForestClassifier(random_state=42)
```

```
RandomForestClassifier
RandomForestClassifier(random_state=42)
```

In [7]:

```
bp = rf_cv.best_params_
click.echo("Best parameters set found:")
click.echo(bp)
Best parameters set found:
{'criterion': 'gini', 'max_depth': 5, 'max_features': 'sqrt',
'n_estimators': 100}
```

In [8]:

```
rf = RandomForestClassifier(random_state=42,
max_features=bp["max_features"], n_estimators=bp["n_estimators"],
max_depth=bp["max_depth"], criterion=bp["criterion"])
```

In [9]:

```
rf.fit(X_train, y_train.values.ravel())
```

Out[9]:

```
RandomForestClassifier
RandomForestClassifier(max_depth=5, random_state=42)
```

In [ ]:

In [10]:

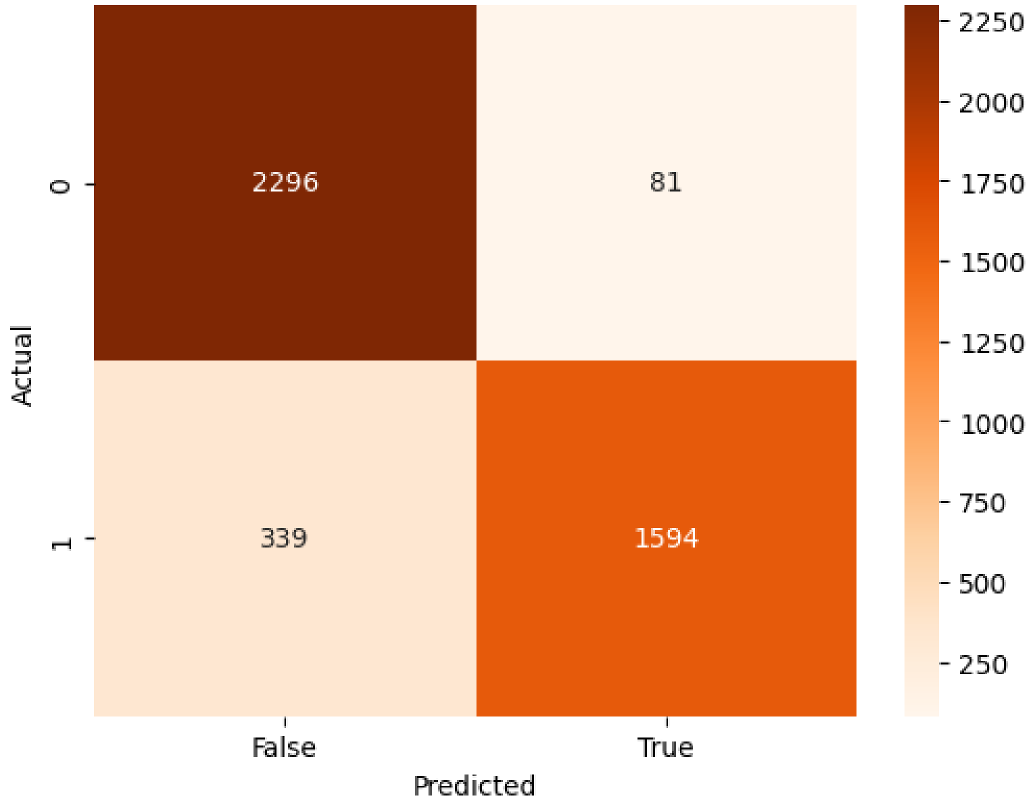
```
# Predict the malicious column using the test data
#add the incepts


y_predicted = rf.predict(X_test)

# Present the results
click.echo("Features selected:")
click.echo(X.columns)
click.echo("Confusion matrix:")
cm = confusion_matrix(y_test, y_predicted)
click.echo(cm)
click.echo("Classification report:")
click.echo(classification_report(y_test, y_predicted))


# Heatmap of confusion matrix
y_predicted

threshold = 0.5
y_predicted = [y > threshold for y in y_predicted]
data = {'Actual':    y_test.values.flatten(),
        'Predicted': y_predicted
        }

# Generate a confusion matrix and heatmap to evaluate the Type I and Type
II errors/ FP/FN etc.
df = pd.DataFrame(data, columns=['Actual','Predicted'])
cm2 = pd.crosstab(df['Actual'], df['Predicted'], rownames=['Actual'],
colnames=['Predicted'])
fig = sns.heatmap(cm2, annot=True, cmap='Oranges', fmt='g')
fig
```

```
Features selected:
Index(['domain_to_earliest_cert_delta', 'ctlog_earliest_dow_sin',
       'ctlog_earliest_dow_cos'],
      dtype='object')
Confusion matrix:
[[2296   81]
 [ 339 1594]]
Classification report:
              precision    recall  f1-score   support

           0       0.87      0.97      0.92      2377
           1       0.95      0.82      0.88      1933

    accuracy                           0.90      4310
```

```
   macro avg       0.91      0.90      0.90      4310
weighted avg       0.91      0.90      0.90      4310
```

Out[10]:

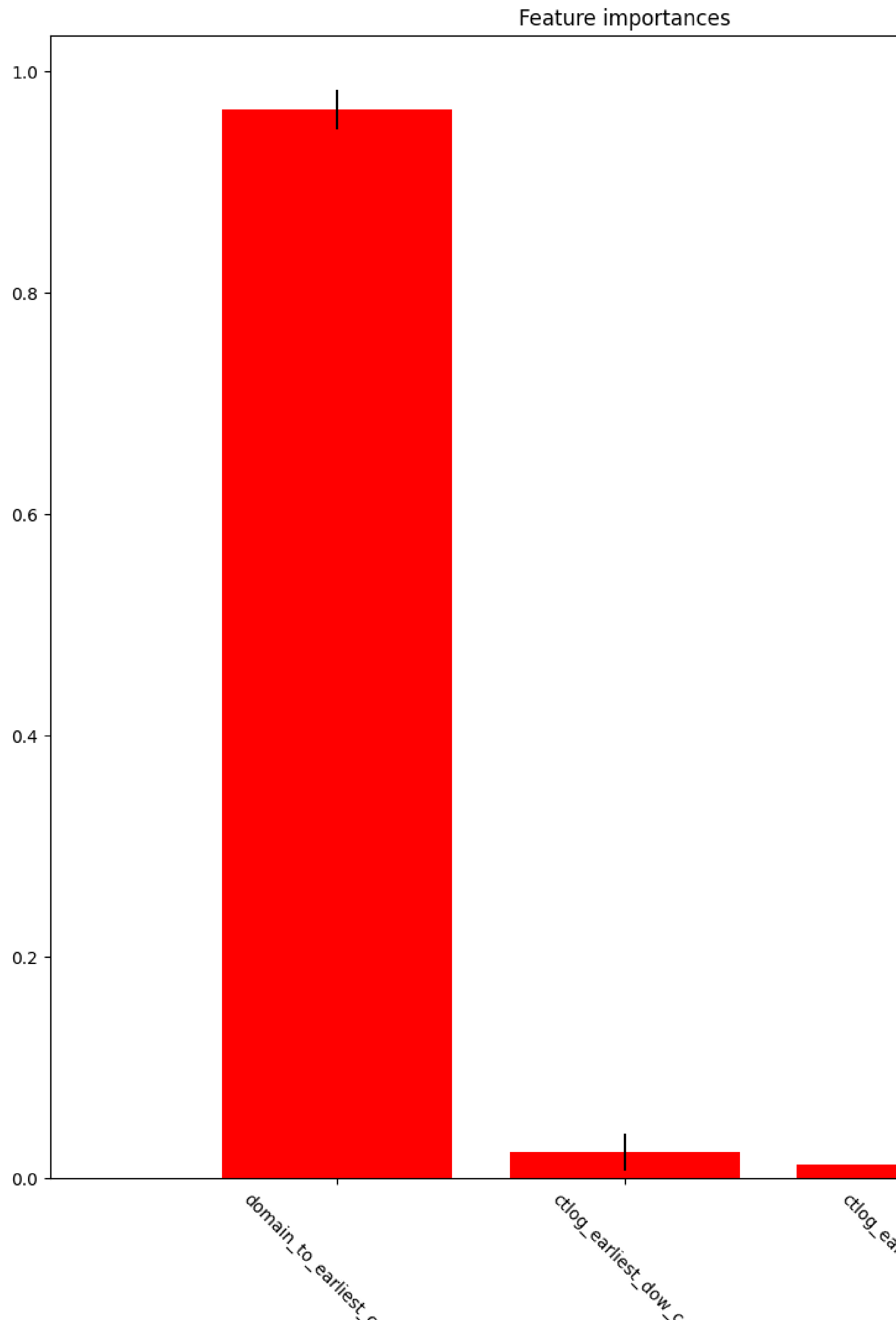
```
<Axes: xlabel='Predicted', ylabel='Actual'>
```



In [11]:

```
# plot the feature importances
importances = rf.feature_importances_
std = np.std([tree.feature_importances_ for tree in rf.estimators_],
axis=0)

indices = np.argsort(importances)[::-1]
# Print the feature ranking
click.echo("Feature ranking:")
for f in range(X.shape[1]):
    click.echo("%d. feature %s (%f)" % (f + 1, combo_features[indices[f]],
importances[indices[f]]))


# Plot the feature importances of the forest
plt.figure(figsize=(12,12))
plt.title("Feature importances")
plt.bar(range(X.shape[1]), importances[indices], color="r",
yerr=std[indices], align="center")
plt.xticks(range(X.shape[1]), X.columns[indices], rotation=-45)
plt.xlim([-1, X.shape[1]])
plt.show()
```

```
Feature ranking:
1. feature domain_to_earliest_cert_delta (0.965421)
2. feature ctlog_earliest_dow_cos (0.022724)
3. feature ctlog_earliest_dow_sin (0.011854)
```



Feature importances
1.0
0.8
0.6
0.4
0.2
0.0

III. Feature Set B

In [1]:

```
import click
import pandas as pd
import glob
import os
#import sklearn
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.metrics import accuracy_score, f1_score, confusion_matrix,
classification_report, roc_auc_score, roc_curve, auc
from sklearn.preprocessing import StandardScaler
from scipy import stats
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
import statsmodels.api as sm
import matplotlib.pyplot as plt
from datetime import datetime, date
# qqplot of the data
import seaborn as sns
import math
import numpy as np
import utils

combo_features =     ['domain_to_earliest_cert_delta', 'ctlog_wildcard']

path = "../data/"
filename = None

epoch = datetime(1970,1,1)
# open the training data file, from ../data/ for the most recent merged
training data file saved by prepare-training-data-parallel.py
full_path = path + "merged_20230705-104357_training_adorned-engineered.csv"

# get latest file matching the glob
if not filename:
    filename = max(glob.glob(full_path), key=os.path.getctime)
    click.echo(f"Using {filename} as training data")
    df = pd.read_csv(filename, sep=",")

# randomize the rows
df = df.sample(frac=1, random_state=42).reset_index(drop=True)

click.echo(df.shape)
click.echo(df.columns)

""" # From prepare-training-data-parallel.py:
# Columns:
url
verification_time
```

```
domain
malicious
dateadded
whois_created
soa_timestamp
ctlog_earliest
ctlog_latest
ctlog_wildcard
whois_created_dayofweek,
ctlog_earliest_dayofweek,
domain_to_cert_delta
"""


# Data cleaning - there may be some epoch values in the whois_created
#  & ct_log_earliest columns, so we need to remove those rows

# convert the whois_created and ctlog_earliest, ctlog_latest columns to
datetime

df = df.loc[df["whois_created"] != ""]
df = df.loc[df["ctlog_earliest"] != ""]
df = df.loc[df["ctlog_latest"] != ""]

# some dates are have nanoseconds in them, so we need to remove the
nanosecond part
df["whois_created"] = df["whois_created"].str.split(".", n = 1, expand =
True)[0]
# some dates has additional timezone information, so we need to remove that
df["whois_created"] = df["whois_created"].apply(lambda x: " ".join(
str(x).split(" ")[:2]))


# convert the whois_created and ct_log_earliest columns to datetime
df = df.astype({"whois_created": 'datetime64[ns]', "ctlog_earliest":
'datetime64[ns]', "ctlog_latest": 'datetime64[ns]'})
df = df.loc[df["whois_created"].ne(pd.Timestamp('1970-01-01 00:00:00'))]
df = df.loc[df["ctlog_earliest"].ne(pd.Timestamp('1970-01-01 00:00:00'))]
df = df.loc[df["ctlog_latest"].ne(pd.Timestamp('1970-01-01 00:00:00'))]


# malicious is a bool
df['malicious'] = df['malicious'].astype('bool')
df['domain'] = df['domain'].astype('string')
Using ../data/merged_20230705-104357_training_adorned-engineered.csv as
training data
(35438, 13)
Index(['url', 'verification_time', 'domain', 'malicious', 'dateadded',
       'whois_created', 'soa_timestamp', 'ctlog_earliest', 'ctlog_latest',
```

```
       'ctlog_wildcard', 'whois_created_dayofweek',
'ctlog_earliest_dayofweek',
       'domain_to_cert_delta'],
      dtype='object')
```

In [2]:

```
#########
# Feature engineering
#########

# remove any rows where the whois_created or ctlog_earliest columns are
null
df = df.loc[df["whois_created"].notnull()]
df = df.loc[df["ctlog_earliest"].notnull()]

# if the data is missing the "whois_created_dayofweek" or
"ctlog_earliest_dayofweek" columns, then add them
# whois created day of the week 0 = Monday, 6 = Sunday
df["whois_created_dayofweek"] = df["whois_created"].apply(
        lambda x: x.weekday() if isinstance(x, date) else None
    )

# cert valid day of the week 0 = Monday, 6 = Sunday
df["ctlog_earliest_dayofweek"] = df["ctlog_earliest"].apply(
        lambda x: x.weekday() if isinstance(x, date) else None
    )

df["ctlog_latest_dayofweek"] = df["ctlog_latest"].apply(
        lambda x: x.weekday() if isinstance(x, date) else None
    )

# set data type of the day of week columns to int
df["whois_created_dayofweek"] =
df["whois_created_dayofweek"].astype("int64")
df["ctlog_earliest_dayofweek"] =
df["ctlog_earliest_dayofweek"].astype("int64")
df["ctlog_latest_dayofweek"] = df["ctlog_latest_dayofweek"].astype("int64")

# domain to cert delta
df["domain_to_earliest_cert_delta"] = df.apply(
        lambda x: utils.get_days_delta(
            x["ctlog_earliest"],
            x["whois_created"]
            if x["whois_created"] and x["ctlog_earliest"]
            else None,
        ),
        axis=1,
    )

# set the data type of the domain_to_cert_delta column to float
```

```
df["domain_to_earliest_cert_delta"] =
df["domain_to_earliest_cert_delta"].astype("float64")

# domain to latest cert delta
df["domain_to_latest_cert_delta"] = df.apply(
        lambda x: utils.get_days_delta(
            x["ctlog_latest"],
            x["whois_created"]
            if x["whois_created"] and x["ctlog_latest"]
            else None,
        ),
        axis=1,
    )

# set the data type of the domain_to_cert_delta column to float
df["domain_to_latest_cert_delta"] =
df["domain_to_latest_cert_delta"].astype("float64")

# drop columns we imported that we will never use
df = df.drop(columns=["url", "verification_time", "dateadded",
"soa_timestamp","domain_to_cert_delta"])

click.echo(df.head())
click.echo(df.describe(include='all'))
click.echo(df.dtypes)
                            domain  malicious       whois_created
0  i-db5p-cor001.api.p001.1drv.com      False 2013-08-05 18:33:50  \
4       soundcloud-pax.pandora.com      False 1993-12-28 05:00:00
5         joolcomercializadora.com       True 2023-05-22 14:53:50
6        createpdf-asr.acrobat.com      False 1999-03-16 05:00:00
8                          popt.in      False 2016-05-14 16:58:55

       ctlog_earliest        ctlog_latest  ctlog_wildcard
0 2022-01-24 20:01:58 2023-06-08 20:46:06            True  \
4 2022-05-19 00:00:00 2023-06-19 23:59:59            True
5 2022-04-06 22:23:24 2023-09-22 23:59:59           False
6 2022-09-09 00:00:00 2023-10-10 23:59:59            True
8 2023-01-07 20:36:15 2023-08-15 04:16:52           False

   whois_created_dayofweek  ctlog_earliest_dayofweek
ctlog_latest_dayofweek
0                        0                          0
3  \
4                        1                          3
0
5                        0                          2
4
6                        1                          4
1
```

```
8                        5                          5
1

   domain_to_earliest_cert_delta  domain_to_latest_cert_delta
0                        -3095.0                      -3595.0
4                       -10369.0                     -10766.0
5                          410.0                       -124.0
6                        -8578.0                      -8975.0
8                        -2430.0                      -2649.0
                   domain malicious                  whois_created
count               21549     21549                          21549  \
unique              21536         2                            NaN
top     www.mediafire.com     False                            NaN
freq                    2     11739                            NaN
mean                  NaN       NaN  2012-10-03 12:56:32.335050496
min                   NaN       NaN            1986-01-09 00:00:00
25%                   NaN       NaN            2003-05-25 13:35:05
50%                   NaN       NaN            2015-05-07 23:56:05
75%                   NaN       NaN            2023-03-20 15:03:16
max                   NaN       NaN            2023-07-03 08:21:24
std                   NaN       NaN                            NaN

                        ctlog_earliest                   ctlog_latest
count                            21549                          21549  \
unique                             NaN                            NaN
top                                NaN                            NaN
freq                               NaN                            NaN
mean    2022-09-26 15:45:50.943570432  2023-08-14 17:49:06.400900352
min               2021-11-30 05:24:28            2023-01-01 18:42:11
25%               2022-06-24 13:47:12            2023-07-02 08:11:07
50%               2022-10-18 21:00:14            2023-08-21 21:40:11
75%               2022-12-14 00:00:00            2023-09-21 19:41:38
max               2023-06-28 04:36:22            2023-12-31 23:59:59
std                                NaN                            NaN

       ctlog_wildcard  whois_created_dayofweek  ctlog_earliest_dayofweek
count           21549             21549.000000              21549.000000  \
unique              2                      NaN                       NaN
top             False                      NaN                       NaN
freq            13032                      NaN                       NaN
mean              NaN                 2.332823                  2.399462
min               NaN                 0.000000                  0.000000
25%               NaN                 1.000000                  1.000000
50%               NaN                 2.000000                  2.000000
75%               NaN                 4.000000                  4.000000
max               NaN                 6.000000                  6.000000
std               NaN                 1.775043                  1.897252

        ctlog_latest_dayofweek  domain_to_earliest_cert_delta
```

```
count             21549.000000                  21549.000000  \
unique                     NaN                           NaN
top                        NaN                           NaN
freq                       NaN                           NaN
mean                  2.873080                  -3645.602070
min                   0.000000                 -13445.000000
25%                   1.000000                  -7078.000000
50%                   3.000000                  -2637.000000
75%                   5.000000                     69.000000
max                   6.000000                    524.000000
std                   2.057394                   3790.677119

        domain_to_latest_cert_delta
count                  21549.000000
unique                          NaN
top                             NaN
freq                            NaN
mean                   -3967.678222
min                   -13798.000000
25%                    -7421.000000
50%                    -3009.000000
75%                     -144.000000
max                      135.000000
std                     3852.703681
domain                               string[python]
malicious                                      bool
whois_created                        datetime64[ns]
ctlog_earliest                       datetime64[ns]
ctlog_latest                         datetime64[ns]
ctlog_wildcard                                 bool
whois_created_dayofweek                       int64
ctlog_earliest_dayofweek                      int64
ctlog_latest_dayofweek                        int64
domain_to_earliest_cert_delta               float64
domain_to_latest_cert_delta                 float64
dtype: object
```

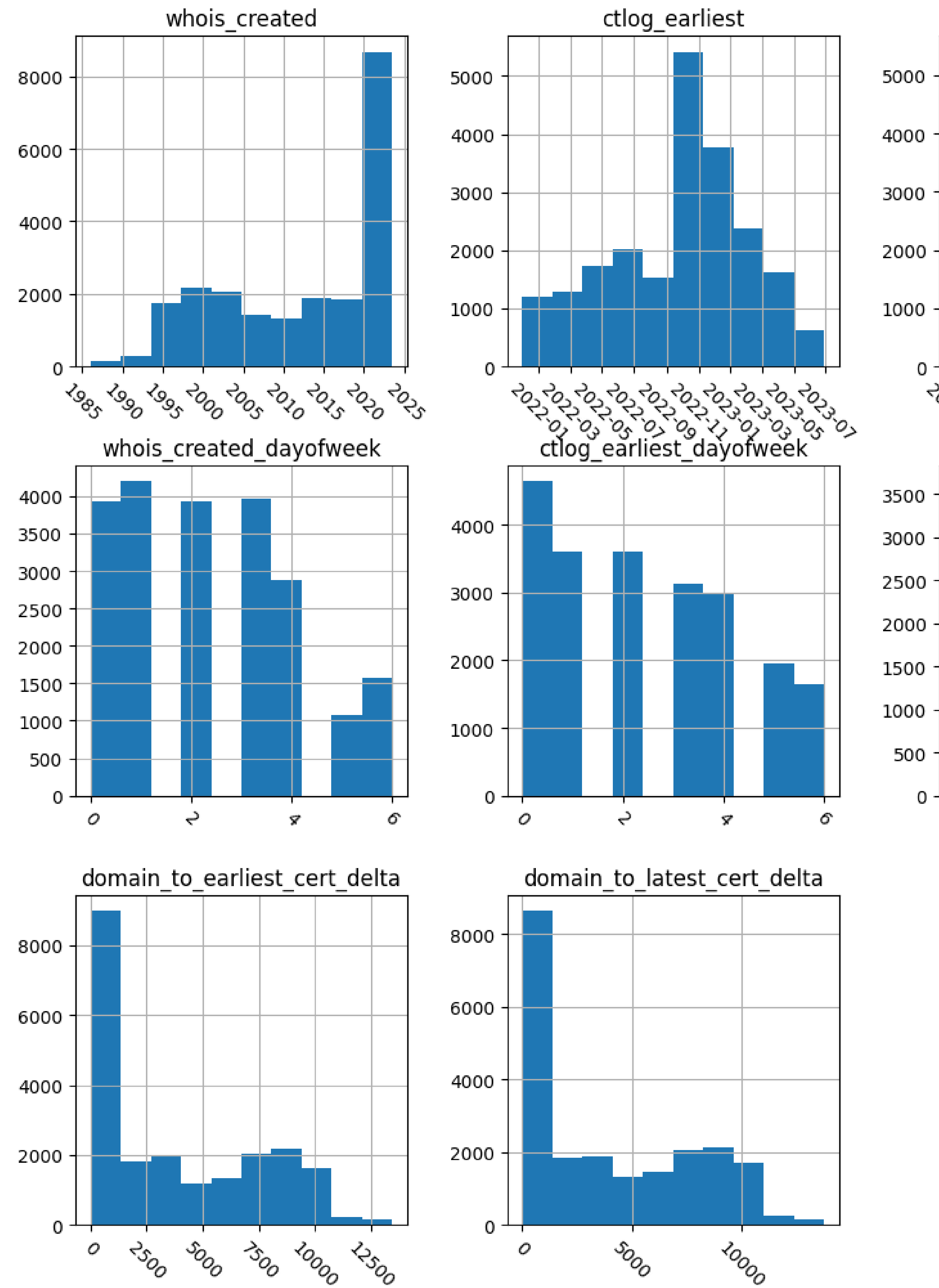
In [3]:

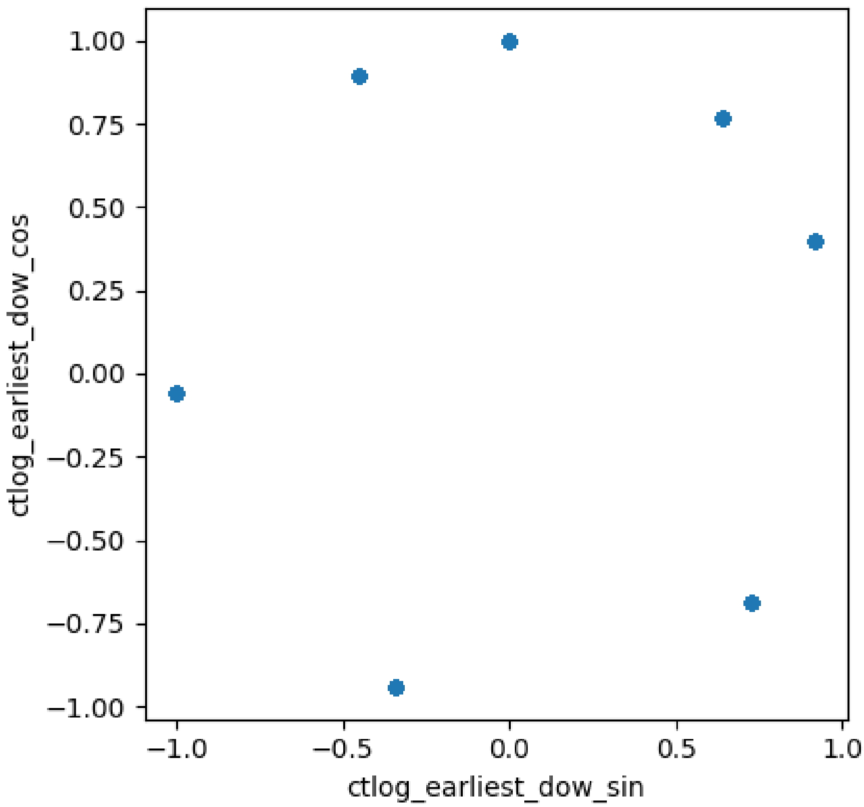
```
# absolute value of the domain_to_cert_delta
df["domain_to_earliest_cert_delta"] =
abs(df["domain_to_earliest_cert_delta"])
df["domain_to_latest_cert_delta"] = abs(df["domain_to_latest_cert_delta"])

df.hist(figsize=(12,12), xrot=-45)

col_index = df.columns.get_loc("whois_created_dayofweek")
df["whois_created_dow_sin"] = df.apply(lambda row:
math.sin(360/7*(row[col_index])),axis=1)
df["whois_created_dow_cos"] = df.apply(lambda row:
math.cos(360/7*(row[col_index])),axis=1)
```

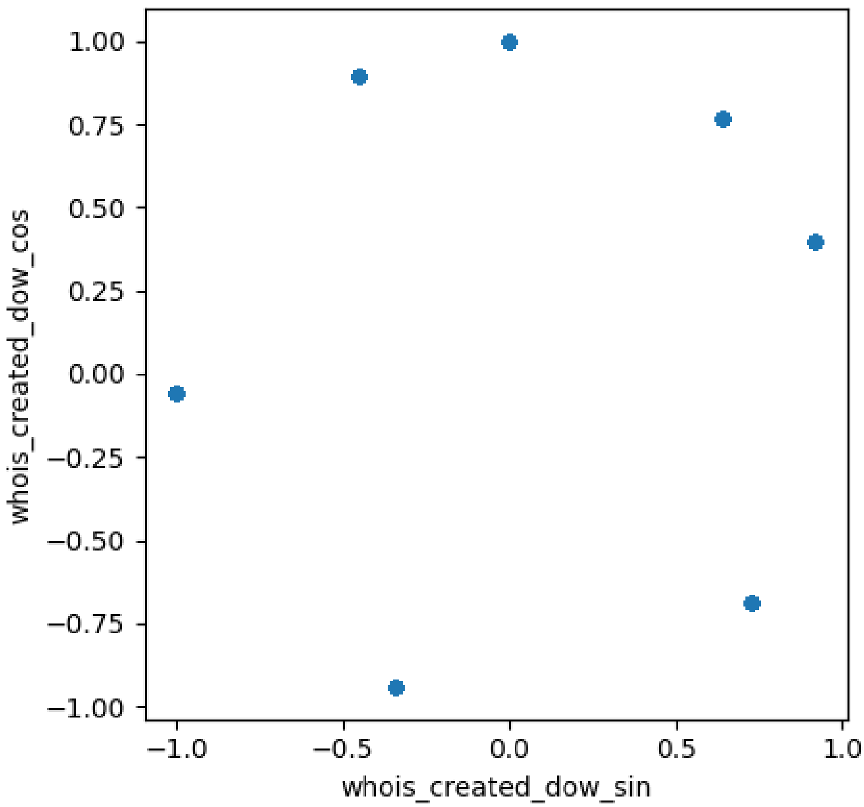
```
col_index = df.columns.get_loc("ctlog_earliest_dayofweek")
df["ctlog_earliest_dow_sin"] = df.apply(lambda row:
math.sin(360/7*(row[col_index])),axis=1)
df["ctlog_earliest_dow_cos"] = df.apply(lambda row:
math.cos(360/7*(row[col_index])),axis=1)

col_index = df.columns.get_loc("ctlog_latest_dayofweek")
df["ctlog_latest_dow_sin"] = df.apply(lambda row:
math.sin(360/7*(row[col_index])),axis=1)
df["ctlog_latest_dow_cos"] = df.apply(lambda row:
math.cos(360/7*(row[col_index])),axis=1)


df.plot.scatter(x="ctlog_earliest_dow_sin",
y="ctlog_earliest_dow_cos").set_aspect('equal')
df.plot.scatter(x="whois_created_dow_sin",
y="whois_created_dow_cos").set_aspect('equal')
df.plot.scatter(x="ctlog_latest_dow_sin",
y="ctlog_latest_dow_cos").set_aspect('equal')

"""
# add one hot encode ctlog_earliest_not_before_dayofweek
df = pd.get_dummies(
        df,
        columns=["ctlog_earliest_dayofweek"],
        prefix=["ctlog_earliest_dow"],
    )
# add one hot encode whois_created_dayofweek to columns
df = pd.get_dummies(
        df, columns=["whois_created_dayofweek"], prefix="whois_dow"
    )
    """

# Summary statistics
click.echo(df.describe(include='all'))
                   domain malicious                  whois_created
count               21549     21549                          21549  \
unique              21536         2                            NaN
top     www.mediafire.com     False                            NaN
freq                    2     11739                            NaN
mean                  NaN       NaN  2012-10-03 12:56:32.335050496
min                   NaN       NaN            1986-01-09 00:00:00
25%                   NaN       NaN            2003-05-25 13:35:05
50%                   NaN       NaN            2015-05-07 23:56:05
75%                   NaN       NaN            2023-03-20 15:03:16
max                   NaN       NaN            2023-07-03 08:21:24
std                   NaN       NaN                            NaN
```

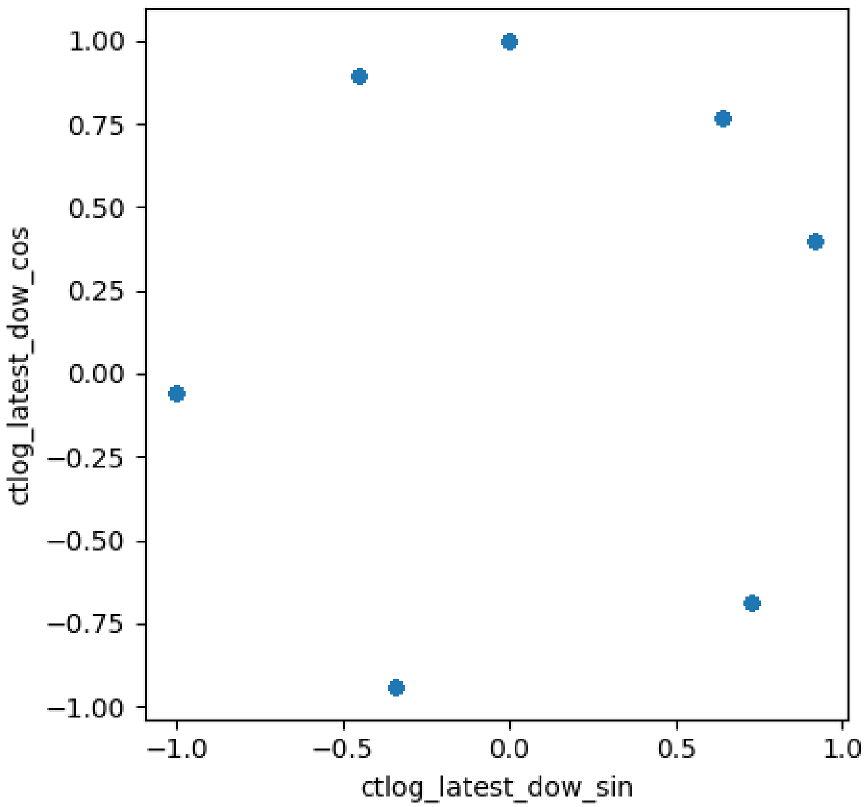
```
                      ctlog_earliest                 ctlog_latest
count                          21549                        21549  \
unique                           NaN                          NaN
top                              NaN                          NaN
freq                             NaN                          NaN
mean    2022-09-26 15:45:50.943570432  2023-08-14 17:49:06.400900352
min              2021-11-30 05:24:28          2023-01-01 18:42:11
25%              2022-06-24 13:47:12          2023-07-02 08:11:07
50%              2022-10-18 21:00:14          2023-08-21 21:40:11
75%              2022-12-14 00:00:00          2023-09-21 19:41:38
max              2023-06-28 04:36:22          2023-12-31 23:59:59
std                              NaN                          NaN

       ctlog_wildcard  whois_created_dayofweek  ctlog_earliest_dayofweek
count           21549             21549.000000              21549.000000  \
unique              2                      NaN                       NaN
top             False                      NaN                       NaN
freq            13032                      NaN                       NaN
mean              NaN                 2.332823                  2.399462
min               NaN                 0.000000                  0.000000
25%               NaN                 1.000000                  1.000000
50%               NaN                 2.000000                  2.000000
75%               NaN                 4.000000                  4.000000
max               NaN                 6.000000                  6.000000
std               NaN                 1.775043                  1.897252

        ctlog_latest_dayofweek  domain_to_earliest_cert_delta
count             21549.000000                   21549.000000  \
unique                     NaN                            NaN
top                        NaN                            NaN
freq                       NaN                            NaN
mean                  2.873080                    3742.948397
min                   0.000000                       0.000000
25%                   1.000000                     181.000000
50%                   3.000000                    2637.000000
75%                   5.000000                    7078.000000
max                   6.000000                   13445.000000
std                   2.057394                    3694.584062

        domain_to_latest_cert_delta  whois_created_dow_sin
count                  21549.000000           21549.000000  \
unique                          NaN                    NaN
top                             NaN                    NaN
freq                            NaN                    NaN
mean                    3969.491206               0.140419
min                        0.000000              -0.998199
25%                      144.000000              -0.340712
50%                     3009.000000               0.000000
75%                     7421.000000               0.728010
```

```
max                   13798.000000               0.918032
std                    3850.835626               0.659922

        whois_created_dow_cos  ctlog_earliest_dow_sin
ctlog_earliest_dow_cos
count            21549.000000            21549.000000
21549.000000  \
unique                    NaN                     NaN
NaN
top                       NaN                     NaN
NaN
freq                      NaN                     NaN
NaN
mean                 0.054288                0.095357
0.161451
min                 -0.940168               -0.998199                -
0.940168
25%                 -0.685567               -0.340712                -
0.685567
50%                  0.396506                0.000000
0.396506
75%                  0.767830                0.728010
0.892589
max                  1.000000                0.918032
1.000000
std                  0.736128                0.651782
0.734891

        ctlog_latest_dow_sin  ctlog_latest_dow_cos
count           21549.000000          21549.000000
unique                   NaN                   NaN
top                      NaN                   NaN
freq                     NaN                   NaN
mean                0.096253              0.255578
min                -0.998199             -0.940168
25%                -0.450871             -0.685567
50%                 0.000000              0.396506
75%                 0.728010              0.892589
max                 0.918032              1.000000
std                 0.651597              0.707728
```


whois_created
ctlog_earliest
whois_created_dayofweek
ctlog_earliest_dayofweek
domain_to_earliest_cert_delta
domain_to_latest_cert_delta
0
2000
4000
6000
8000
1985
1990
1995
2000
2005
2010
2015
2020
2025
1000
3000
5000
2022-01
2022-03
2022-05
2022-07
2022-09
2022-11
2023-01
2023-03
2023-05
2023-07
500
1500
2500
3500
2
4
6
7500
10000
12500


1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
ctlog_earliest_dow_cos
−1.0
−0.5
0.0
0.5
1.0
ctlog_earliest_dow_sin

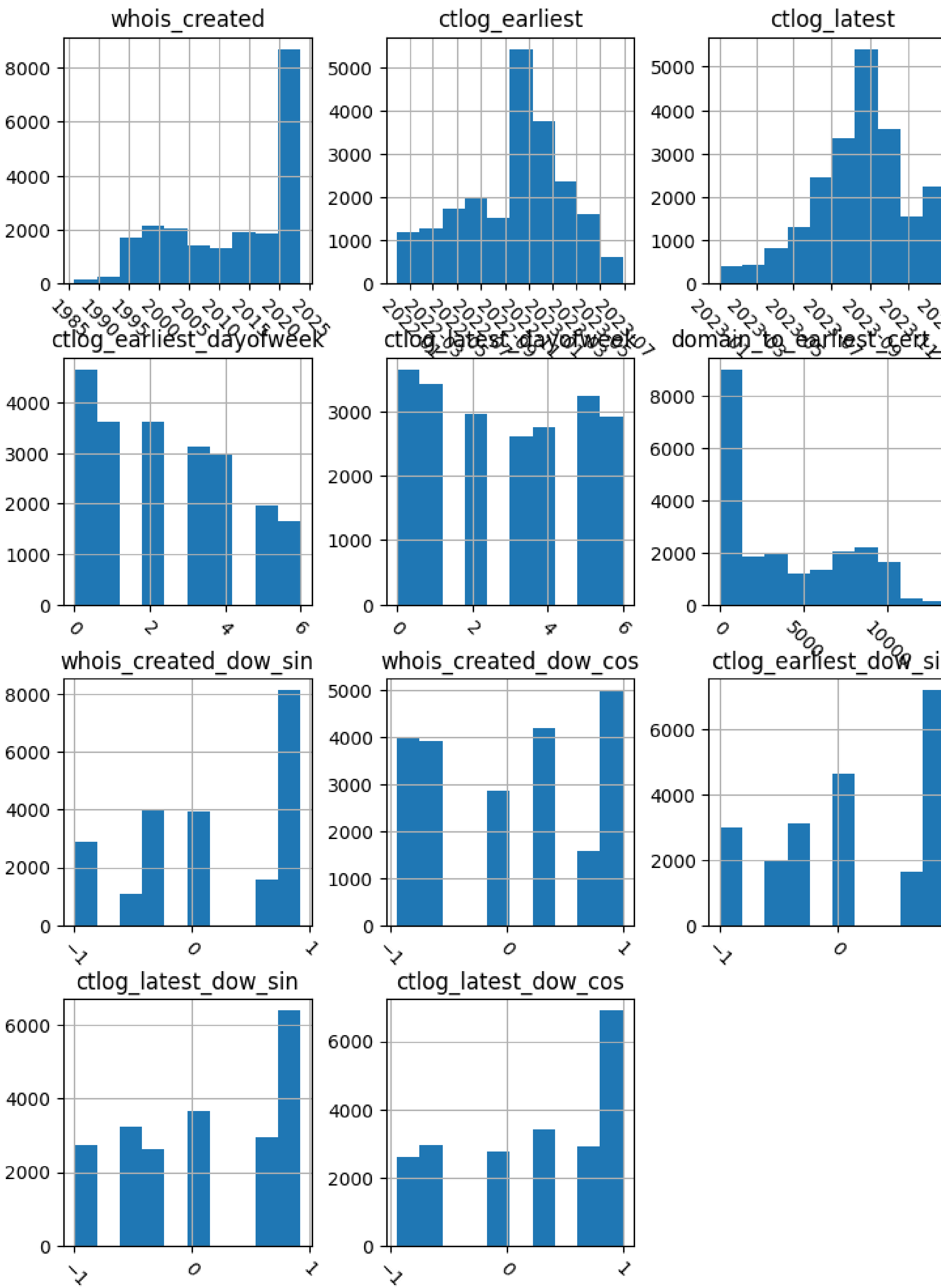

1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
whois_created_dow_cos
−1.0
−0.5
0.0
0.5
1.0
whois_created_dow_sin

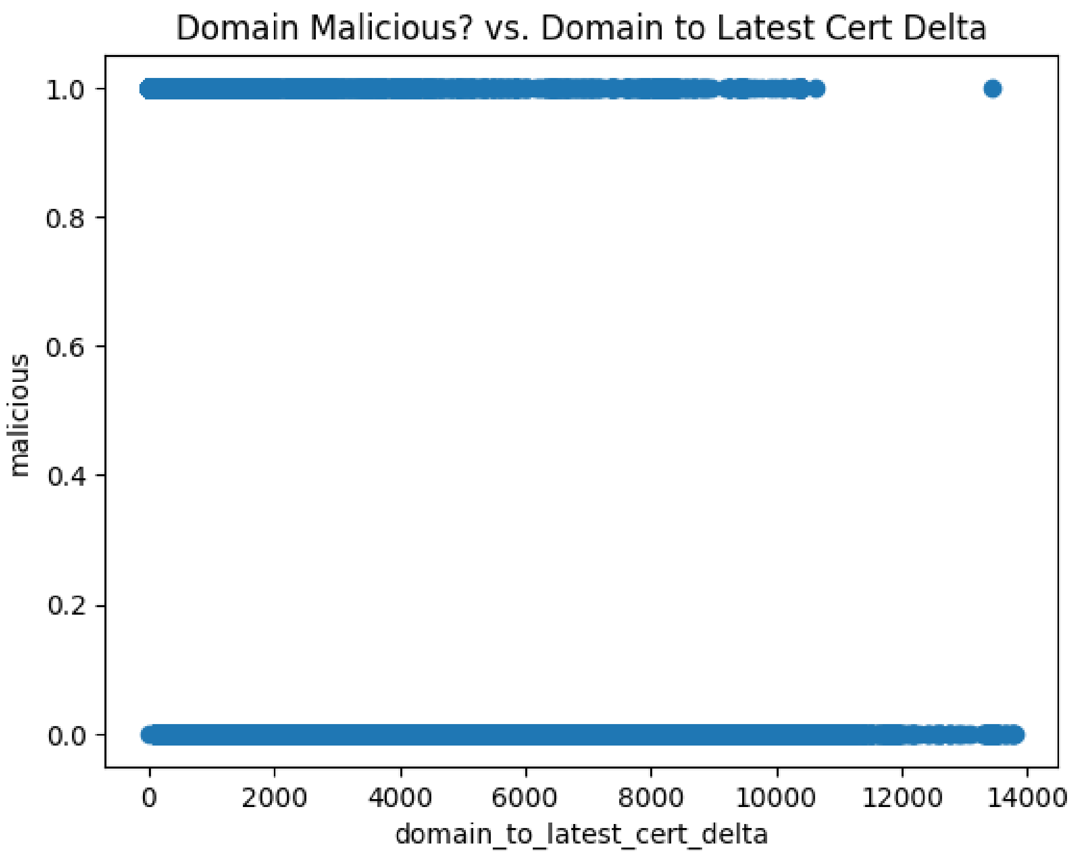


In [4]:

```
# Plot histograms
df.hist(figsize=(12,12), xrot=-45)

# Create a scatter plot of the data, showing malicious and benign domains
# plot the data as a scatter plot
plt.scatter(df["domain_to_earliest_cert_delta"], df["malicious"])
plt.xlabel("domain_to_earliest_cert_delta")
plt.ylabel("malicious")
plt.title("Domain Malicious? vs. Domain to Earliest Cert Delta")
plt.show()
# and for the latest
plt.scatter(df["domain_to_latest_cert_delta"], df["malicious"])
plt.xlabel("domain_to_latest_cert_delta")
plt.ylabel("malicious")
plt.title("Domain Malicious? vs. Domain to Latest Cert Delta")
plt.show()



click.echo(df.head())

# print a count of malicious and benign values
click.echo(df["malicious"].value_counts())
# deduplicate any rows, there shouldn't be any, but just in case
df = df.drop_duplicates()
click.echo(df["malicious"].value_counts())
```

```
click.echo(df.head())

X = df.drop(["malicious","domain", "ctlog_earliest", "ctlog_latest",
"whois_created", "whois_created_dayofweek", "ctlog_earliest_dayofweek"],
axis=1)
X = df.filter(combo_features)
y = df.filter(["malicious"])


click.echo(X.describe())
```


whois_created
ctlog_earliest
ctlog_latest
ctlog_earliest_dayofweek
ctlog_latest_dayofweek
whois_created_dow_sin
whois_created_dow_cos
ctlog_latest_dow_sin
ctlog_latest_dow_cos




```
                             domain  malicious        whois_created
0  i-db5p-cor001.api.p001.1drv.com      False 2013-08-05 18:33:50   \
4       soundcloud-pax.pandora.com      False 1993-12-28 05:00:00
5         joolcomercializadora.com       True 2023-05-22 14:53:50
6        createpdf-asr.acrobat.com      False 1999-03-16 05:00:00
8                          popt.in      False 2016-05-14 16:58:55

       ctlog_earliest         ctlog_latest  ctlog_wildcard
0 2022-01-24 20:01:58 2023-06-08 20:46:06            True   \
4 2022-05-19 00:00:00 2023-06-19 23:59:59            True
5 2022-04-06 22:23:24 2023-09-22 23:59:59           False
6 2022-09-09 00:00:00 2023-10-10 23:59:59            True
8 2023-01-07 20:36:15 2023-08-15 04:16:52           False

   whois_created_dayofweek  ctlog_earliest_dayofweek
ctlog_latest_dayofweek
0                        0                          0
3  \
4                        1                          3
0
5                        0                          2
4
6                        1                          4
1
8                        5                          5
1
```

```
   domain_to_earliest_cert_delta  domain_to_latest_cert_delta
0                         3095.0                       3595.0  \
4                        10369.0                      10766.0
5                          410.0                        124.0
6                         8578.0                       8975.0
8                         2430.0                       2649.0

   whois_created_dow_sin  whois_created_dow_cos  ctlog_earliest_dow_sin
0               0.000000               1.000000                0.000000  \
4               0.918032               0.396506               -0.340712
5               0.000000               1.000000                0.728010
6               0.918032               0.396506               -0.998199
8              -0.450871               0.892589               -0.450871

   ctlog_earliest_dow_cos  ctlog_latest_dow_sin  ctlog_latest_dow_cos
0                1.000000             -0.340712             -0.940168
4               -0.940168              0.000000              1.000000
5               -0.685567             -0.998199             -0.059997
6               -0.059997              0.918032              0.396506
8                0.892589              0.918032              0.396506
malicious
False    11739
True      9810
Name: count, dtype: int64
malicious
False    11739
True      9810
Name: count, dtype: int64
                            domain  malicious       whois_created
0  i-db5p-cor001.api.p001.1drv.com      False 2013-08-05 18:33:50  \
4       soundcloud-pax.pandora.com      False 1993-12-28 05:00:00
5         joolcomercializadora.com       True 2023-05-22 14:53:50
6        createpdf-asr.acrobat.com      False 1999-03-16 05:00:00
8                          popt.in      False 2016-05-14 16:58:55

       ctlog_earliest        ctlog_latest  ctlog_wildcard
0 2022-01-24 20:01:58 2023-06-08 20:46:06            True  \
4 2022-05-19 00:00:00 2023-06-19 23:59:59            True
5 2022-04-06 22:23:24 2023-09-22 23:59:59           False
6 2022-09-09 00:00:00 2023-10-10 23:59:59            True
8 2023-01-07 20:36:15 2023-08-15 04:16:52           False

   whois_created_dayofweek  ctlog_earliest_dayofweek
ctlog_latest_dayofweek
0                        0                         0
3  \
4                        1                         3
0
```

```
5                        0                          2
4
6                        1                          4
1
8                        5                          5
1

   domain_to_earliest_cert_delta  domain_to_latest_cert_delta
0                         3095.0                       3595.0  \
4                        10369.0                      10766.0
5                          410.0                        124.0
6                         8578.0                       8975.0
8                         2430.0                       2649.0

   whois_created_dow_sin  whois_created_dow_cos  ctlog_earliest_dow_sin
0               0.000000               1.000000                0.000000  \
4               0.918032               0.396506               -0.340712
5               0.000000               1.000000                0.728010
6               0.918032               0.396506               -0.998199
8              -0.450871               0.892589               -0.450871

   ctlog_earliest_dow_cos  ctlog_latest_dow_sin  ctlog_latest_dow_cos
0                1.000000             -0.340712             -0.940168
4               -0.940168              0.000000              1.000000
5               -0.685567             -0.998199             -0.059997
6               -0.059997              0.918032              0.396506
8                0.892589              0.918032              0.396506
       domain_to_earliest_cert_delta
count                   21549.000000
mean                     3742.948397
std                      3694.584062
min                         0.000000
25%                       181.000000
50%                      2637.000000
75%                      7078.000000
max                     13445.000000
```

In [5]:

```
# convert y (malicious) to 1/0 int
y = y.astype('int')
if "ctlog_wildcard" in X.columns:
    X["ctlog_wildcard"] = X["ctlog_wildcard"].astype('int')

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)


# random forest model

param_grid = {
```

```
    'n_estimators': [50,100,150,200],
    'max_features': ['sqrt', 'log2'],
    'max_depth' : [2,3,4,5],
    'criterion' :['gini', 'entropy']
}
```

In [6]:

```
rf = RandomForestClassifier(random_state=42)
rf_cv = GridSearchCV(estimator=rf, param_grid=param_grid, cv=5, n_jobs=-1)
rf_cv.fit(X_train, y_train.values.ravel())
```

Out[6]:

GridSearchCV

```
GridSearchCV(cv=5, estimator=RandomForestClassifier(random_state=42),
n_jobs=-1,
             param_grid={'criterion': ['gini', 'entropy'],
                         'max_depth': [2, 3, 4, 5],
                         'max_features': ['sqrt', 'log2'],
                         'n_estimators': [50, 100, 150, 200]})
```

estimator: RandomForestClassifier

```
RandomForestClassifier(random_state=42)
```

```
RandomForestClassifier
RandomForestClassifier(random_state=42)
```

In [7]:

```
bp = rf_cv.best_params_
click.echo("Best parameters set found:")
click.echo(bp)
Best parameters set found:
{'criterion': 'gini', 'max_depth': 5, 'max_features': 'sqrt',
'n_estimators': 150}
```

In [8]:

```
rf = RandomForestClassifier(random_state=42,
max_features=bp["max_features"], n_estimators=bp["n_estimators"],
max_depth=bp["max_depth"], criterion=bp["criterion"])
```

In [9]:

```
rf.fit(X_train, y_train.values.ravel())
```

Out[9]:

```
RandomForestClassifier
RandomForestClassifier(max_depth=5, n_estimators=150, random_state=42)
```

In [ ]:

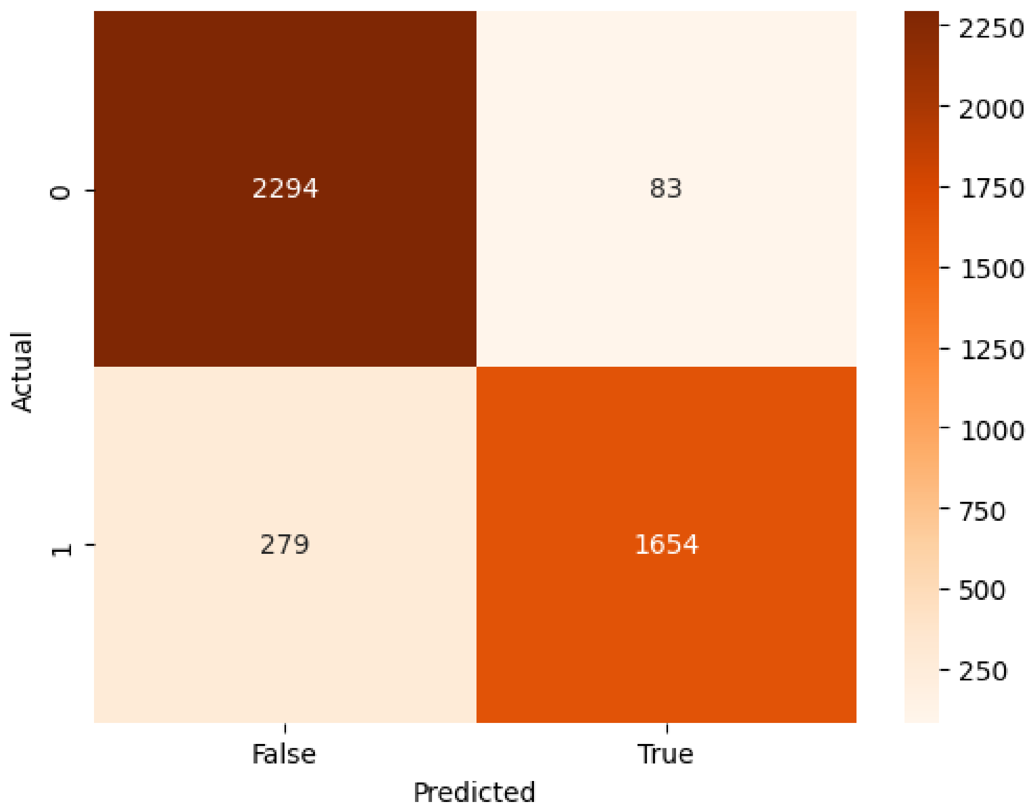
In [10]:

```
# Predict the malicious column using the test data
#add the incepts


y_predicted = rf.predict(X_test)

# Present the results
click.echo("Features selected:")
click.echo(X.columns)
```

```
click.echo("Confusion matrix:")
cm = confusion_matrix(y_test, y_predicted)
click.echo(cm)
click.echo("Classification report:")
click.echo(classification_report(y_test, y_predicted))


# Heatmap of confusion matrix
y_predicted

threshold = 0.5
y_predicted = [y > threshold for y in y_predicted]
data = {'Actual':    y_test.values.flatten(),
        'Predicted': y_predicted
        }

# Generate a confusion matrix and heatmap to evaluate the Type I and Type
II errors/ FP/FN etc.
df = pd.DataFrame(data, columns=['Actual','Predicted'])
cm2 = pd.crosstab(df['Actual'], df['Predicted'], rownames=['Actual'],
colnames=['Predicted'])
fig = sns.heatmap(cm2, annot=True, cmap='Oranges', fmt='g')
fig
Features selected:
Index(['domain_to_earliest_cert_delta', 'ctlog_wildcard'], dtype='object')
Confusion matrix:
[[2294   83]
 [ 279 1654]]
Classification report:
              precision    recall  f1-score   support

           0       0.89      0.97      0.93      2377
           1       0.95      0.86      0.90      1933

    accuracy                           0.92      4310
   macro avg       0.92      0.91      0.91      4310
weighted avg       0.92      0.92      0.92      4310
```

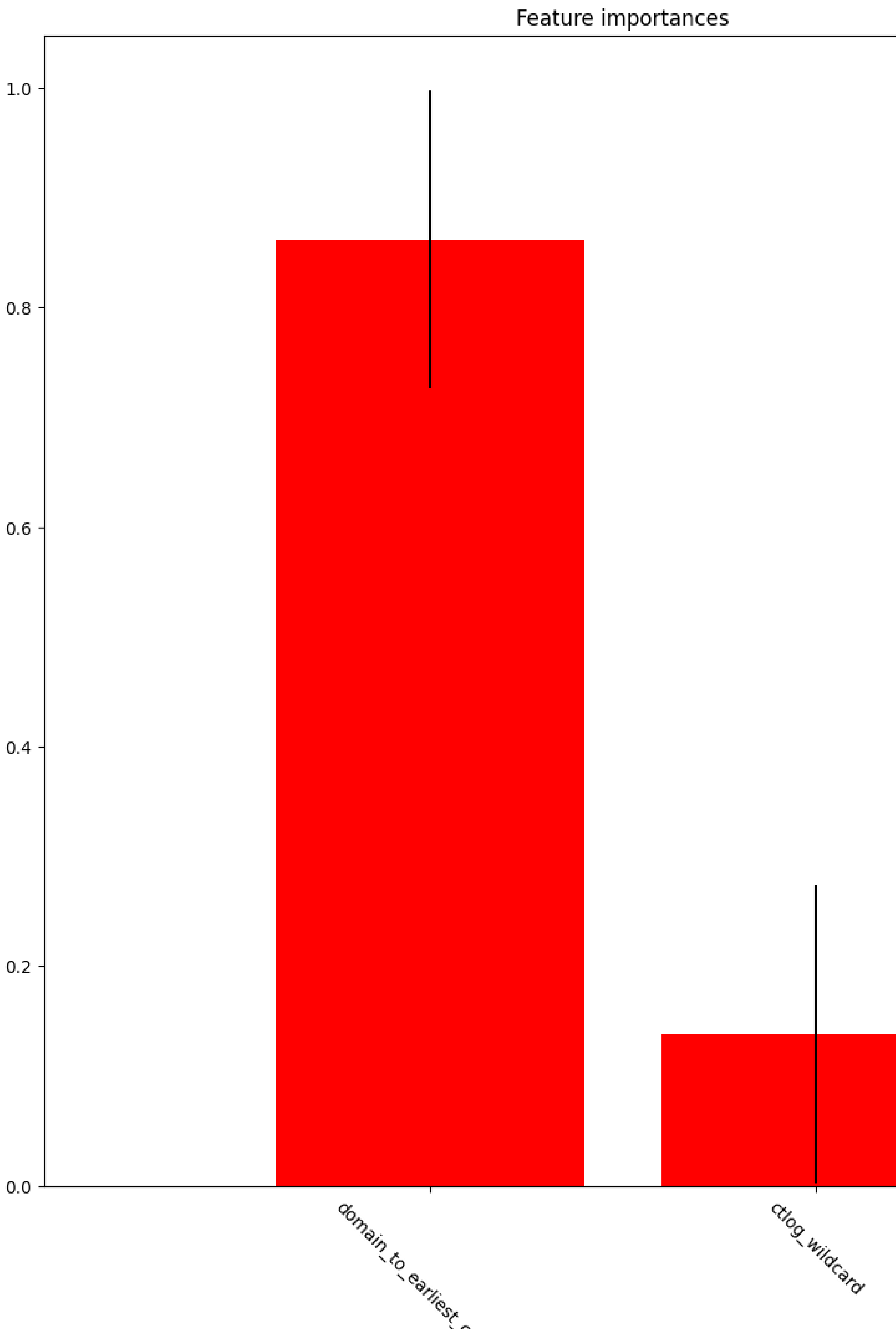
Out[10]:

```
<Axes: xlabel='Predicted', ylabel='Actual'>
```



In [11]:

```
# plot the feature importances
importances = rf.feature_importances_
std = np.std([tree.feature_importances_ for tree in rf.estimators_],
axis=0)

indices = np.argsort(importances)[::-1]
# Print the feature ranking
click.echo("Feature ranking:")
for f in range(X.shape[1]):
    click.echo("%d. feature %s (%f)" % (f + 1, combo_features[indices[f]],
importances[indices[f]]))

# Plot the feature importances of the forest
plt.figure(figsize=(12,12))
plt.title("Feature importances")
plt.bar(range(X.shape[1]), importances[indices], color="r",
yerr=std[indices], align="center")
plt.xticks(range(X.shape[1]), X.columns[indices], rotation=-45)
plt.xlim([-1, X.shape[1]])
plt.show()
Feature ranking:
1. feature domain_to_earliest_cert_delta (0.861852)
2. feature ctlog_wildcard (0.138148)
```



Feature importances
1.0
0.8
0.6
0.4
0.2
0.0
ctlog_wildcard

IV. Feature Set C

In [1]:

```
import click
import pandas as pd
import glob
import os
#import sklearn
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.metrics import accuracy_score, f1_score, confusion_matrix,
classification_report, roc_auc_score, roc_curve, auc
from sklearn.preprocessing import StandardScaler
from scipy import stats
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
import statsmodels.api as sm
import matplotlib.pyplot as plt
from datetime import datetime, date
# qqplot of the data
import seaborn as sns
import math
import numpy as np
import utils

combo_features = [
    'domain_to_earliest_cert_delta',
    'ctlog_earliest_dow_sin',
    'ctlog_earliest_dow_cos',
    'ctlog_wildcard'
    ]

path = "../data/"
filename = None

epoch = datetime(1970,1,1)
# open the training data file, from ../data/ for the most recent merged
training data file saved by prepare-training-data-parallel.py
full_path = path + "merged_20230705-104357_training_adorned-engineered.csv"

# get latest file matching the glob
if not filename:
    filename = max(glob.glob(full_path), key=os.path.getctime)
    click.echo(f"Using {filename} as training data")
    df = pd.read_csv(filename, sep=",")

# randomize the rows
df = df.sample(frac=1, random_state=42).reset_index(drop=True)

click.echo(df.shape)
click.echo(df.columns)
```

```
""" # From prepare-training-data-parallel.py:
# Columns:
url
verification_time
domain
malicious
dateadded
whois_created
soa_timestamp
ctlog_earliest
ctlog_latest
ctlog_wildcard
whois_created_dayofweek,
ctlog_earliest_dayofweek,
domain_to_cert_delta
"""


# Data cleaning - there may be some epoch values in the whois_created
#  & ct_log_earliest columns, so we need to remove those rows

# convert the whois_created and ctlog_earliest, ctlog_latest columns to
datetime

df = df.loc[df["whois_created"] != ""]
df = df.loc[df["ctlog_earliest"] != ""]
df = df.loc[df["ctlog_latest"] != ""]

# some dates are have nanoseconds in them, so we need to remove the
nanosecond part
df["whois_created"] = df["whois_created"].str.split(".", n = 1, expand =
True)[0]
# some dates has additional timezone information, so we need to remove that
df["whois_created"] = df["whois_created"].apply(lambda x: " ".join(
str(x).split(" ")[:2]))


# convert the whois_created and ct_log_earliest columns to datetime
df = df.astype({"whois_created": 'datetime64[ns]', "ctlog_earliest":
'datetime64[ns]', "ctlog_latest": 'datetime64[ns]'})
df = df.loc[df["whois_created"].ne(pd.Timestamp('1970-01-01 00:00:00'))]
df = df.loc[df["ctlog_earliest"].ne(pd.Timestamp('1970-01-01 00:00:00'))]
df = df.loc[df["ctlog_latest"].ne(pd.Timestamp('1970-01-01 00:00:00'))]


# malicious is a bool
df['malicious'] = df['malicious'].astype('bool')
df['domain'] = df['domain'].astype('string')
```

```
Using ../data/merged_20230705-104357_training_adorned-engineered.csv as
training data
(35438, 13)
Index(['url', 'verification_time', 'domain', 'malicious', 'dateadded',
       'whois_created', 'soa_timestamp', 'ctlog_earliest', 'ctlog_latest',
       'ctlog_wildcard', 'whois_created_dayofweek',
'ctlog_earliest_dayofweek',
       'domain_to_cert_delta'],
      dtype='object')
```

In [2]:

```
#########
# Feature engineering
#########

# remove any rows where the whois_created or ctlog_earliest columns are
null
df = df.loc[df["whois_created"].notnull()]
df = df.loc[df["ctlog_earliest"].notnull()]

# if the data is missing the "whois_created_dayofweek" or
"ctlog_earliest_dayofweek" columns, then add them
# whois created day of the week 0 = Monday, 6 = Sunday
df["whois_created_dayofweek"] = df["whois_created"].apply(
        lambda x: x.weekday() if isinstance(x, date) else None
    )

# cert valid day of the week 0 = Monday, 6 = Sunday
df["ctlog_earliest_dayofweek"] = df["ctlog_earliest"].apply(
        lambda x: x.weekday() if isinstance(x, date) else None
    )

df["ctlog_latest_dayofweek"] = df["ctlog_latest"].apply(
        lambda x: x.weekday() if isinstance(x, date) else None
    )

# set data type of the day of week columns to int
df["whois_created_dayofweek"] =
df["whois_created_dayofweek"].astype("int64")
df["ctlog_earliest_dayofweek"] =
df["ctlog_earliest_dayofweek"].astype("int64")
df["ctlog_latest_dayofweek"] = df["ctlog_latest_dayofweek"].astype("int64")

# domain to cert delta
df["domain_to_earliest_cert_delta"] = df.apply(
        lambda x: utils.get_days_delta(
            x["ctlog_earliest"],
            x["whois_created"]
            if x["whois_created"] and x["ctlog_earliest"]
            else None,
```

```
        ),
        axis=1,
    )

# set the data type of the domain_to_cert_delta column to float
df["domain_to_earliest_cert_delta"] =
df["domain_to_earliest_cert_delta"].astype("float64")

# domain to latest cert delta
df["domain_to_latest_cert_delta"] = df.apply(
        lambda x: utils.get_days_delta(
            x["ctlog_latest"],
            x["whois_created"]
            if x["whois_created"] and x["ctlog_latest"]
            else None,
        ),
        axis=1,
    )

# set the data type of the domain_to_cert_delta column to float
df["domain_to_latest_cert_delta"] =
df["domain_to_latest_cert_delta"].astype("float64")

# drop columns we imported that we will never use
df = df.drop(columns=["url", "verification_time", "dateadded",
"soa_timestamp","domain_to_cert_delta"])

click.echo(df.head())
click.echo(df.describe(include='all'))
click.echo(df.dtypes)
                           domain  malicious       whois_created
0  i-db5p-cor001.api.p001.1drv.com      False 2013-08-05 18:33:50  \
4       soundcloud-pax.pandora.com      False 1993-12-28 05:00:00
5         joolcomercializadora.com       True 2023-05-22 14:53:50
6        createpdf-asr.acrobat.com      False 1999-03-16 05:00:00
8                          popt.in      False 2016-05-14 16:58:55

       ctlog_earliest        ctlog_latest  ctlog_wildcard
0 2022-01-24 20:01:58 2023-06-08 20:46:06            True  \
4 2022-05-19 00:00:00 2023-06-19 23:59:59            True
5 2022-04-06 22:23:24 2023-09-22 23:59:59           False
6 2022-09-09 00:00:00 2023-10-10 23:59:59            True
8 2023-01-07 20:36:15 2023-08-15 04:16:52           False

   whois_created_dayofweek  ctlog_earliest_dayofweek
ctlog_latest_dayofweek
0                        0                         0
3  \
```

```
4                        1                         3
0
5                        0                         2
4
6                        1                         4
1
8                        5                         5
1

   domain_to_earliest_cert_delta  domain_to_latest_cert_delta
0                        -3095.0                      -3595.0
4                       -10369.0                     -10766.0
5                          410.0                       -124.0
6                        -8578.0                      -8975.0
8                        -2430.0                      -2649.0
                   domain malicious                  whois_created
count               21549     21549                          21549  \
unique              21536         2                            NaN
top     www.mediafire.com     False                            NaN
freq                    2     11739                            NaN
mean                  NaN       NaN  2012-10-03 12:56:32.335050496
min                   NaN       NaN            1986-01-09 00:00:00
25%                   NaN       NaN            2003-05-25 13:35:05
50%                   NaN       NaN            2015-05-07 23:56:05
75%                   NaN       NaN            2023-03-20 15:03:16
max                   NaN       NaN            2023-07-03 08:21:24
std                   NaN       NaN                            NaN

                       ctlog_earliest                   ctlog_latest
count                           21549                          21549  \
unique                            NaN                            NaN
top                               NaN                            NaN
freq                              NaN                            NaN
mean    2022-09-26 15:45:50.943570432  2023-08-14 17:49:06.400900352
min               2021-11-30 05:24:28            2023-01-01 18:42:11
25%               2022-06-24 13:47:12            2023-07-02 08:11:07
50%               2022-10-18 21:00:14            2023-08-21 21:40:11
75%               2022-12-14 00:00:00            2023-09-21 19:41:38
max               2023-06-28 04:36:22            2023-12-31 23:59:59
std                               NaN                            NaN

       ctlog_wildcard  whois_created_dayofweek  ctlog_earliest_dayofweek
count           21549             21549.000000              21549.000000  \
unique              2                      NaN                       NaN
top             False                      NaN                       NaN
freq            13032                      NaN                       NaN
mean              NaN                 2.332823                  2.399462
min               NaN                 0.000000                  0.000000
25%               NaN                 1.000000                  1.000000
```

```
50%                NaN                 2.000000                  2.000000
75%                NaN                 4.000000                  4.000000
max                NaN                 6.000000                  6.000000
std                NaN                 1.775043                  1.897252

        ctlog_latest_dayofweek  domain_to_earliest_cert_delta
count             21549.000000                   21549.000000  \
unique                     NaN                            NaN
top                        NaN                            NaN
freq                       NaN                            NaN
mean                  2.873080                   -3645.602070
min                   0.000000                  -13445.000000
25%                   1.000000                   -7078.000000
50%                   3.000000                   -2637.000000
75%                   5.000000                      69.000000
max                   6.000000                     524.000000
std                   2.057394                    3790.677119

        domain_to_latest_cert_delta
count                  21549.000000
unique                          NaN
top                             NaN
freq                            NaN
mean                   -3967.678222
min                   -13798.000000
25%                    -7421.000000
50%                    -3009.000000
75%                     -144.000000
max                      135.000000
std                     3852.703681
domain                             string[python]
malicious                                    bool
whois_created                      datetime64[ns]
ctlog_earliest                     datetime64[ns]
ctlog_latest                       datetime64[ns]
ctlog_wildcard                               bool
whois_created_dayofweek                     int64
ctlog_earliest_dayofweek                    int64
ctlog_latest_dayofweek                      int64
domain_to_earliest_cert_delta             float64
domain_to_latest_cert_delta               float64
dtype: object
```

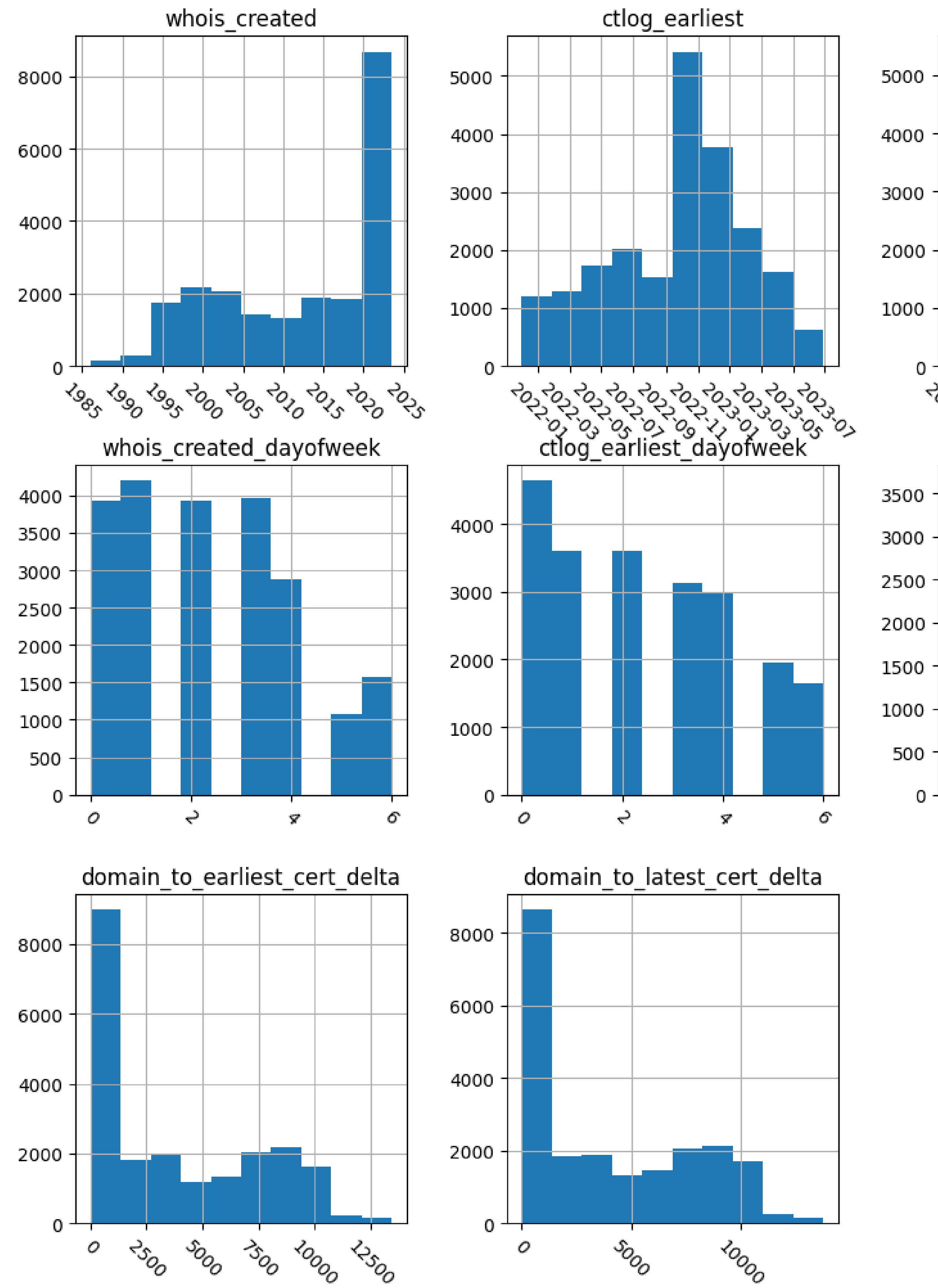
In [3]:

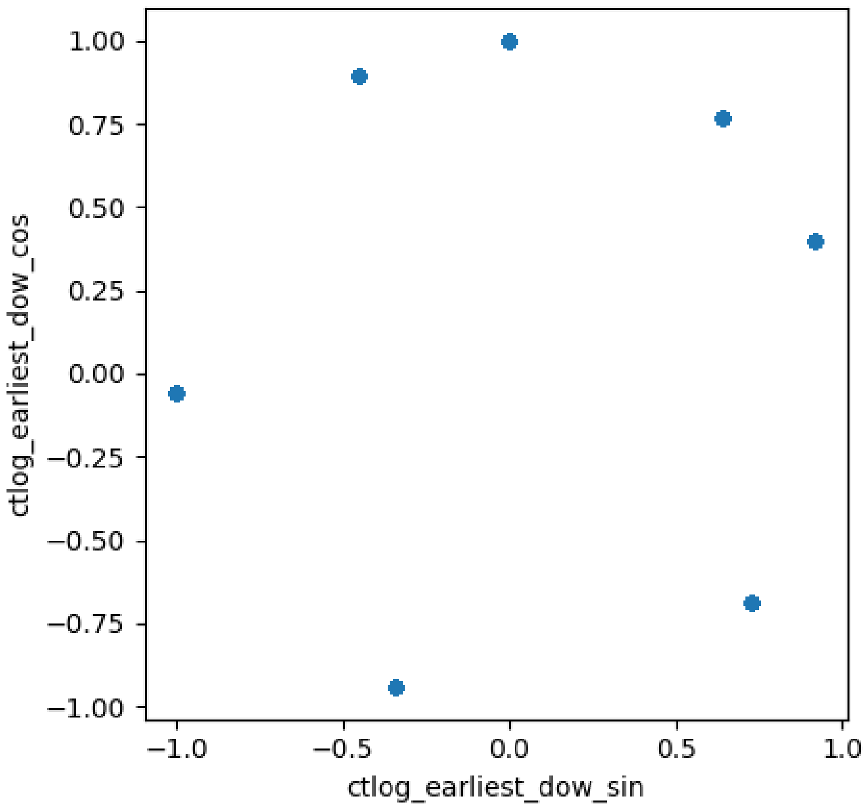
```
# absolute value of the domain_to_cert_delta
df["domain_to_earliest_cert_delta"] =
abs(df["domain_to_earliest_cert_delta"])
df["domain_to_latest_cert_delta"] = abs(df["domain_to_latest_cert_delta"])

df.hist(figsize=(12,12), xrot=-45)
```

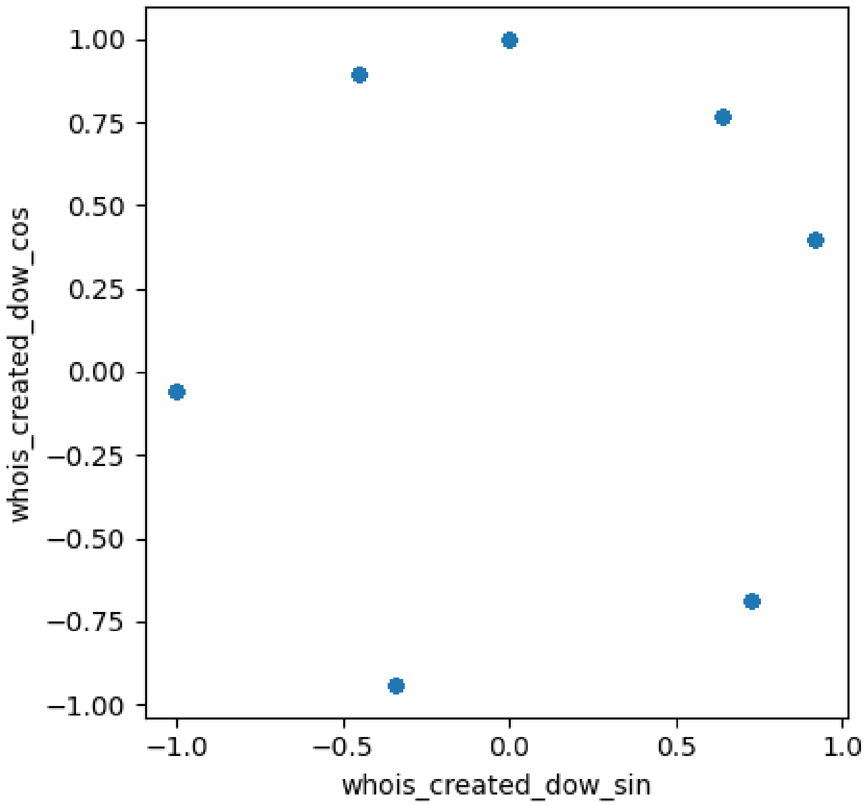
```
col_index = df.columns.get_loc("whois_created_dayofweek")
df["whois_created_dow_sin"] = df.apply(lambda row:
math.sin(360/7*(row[col_index])),axis=1)
df["whois_created_dow_cos"] = df.apply(lambda row:
math.cos(360/7*(row[col_index])),axis=1)

col_index = df.columns.get_loc("ctlog_earliest_dayofweek")
df["ctlog_earliest_dow_sin"] = df.apply(lambda row:
math.sin(360/7*(row[col_index])),axis=1)
df["ctlog_earliest_dow_cos"] = df.apply(lambda row:
math.cos(360/7*(row[col_index])),axis=1)

col_index = df.columns.get_loc("ctlog_latest_dayofweek")
df["ctlog_latest_dow_sin"] = df.apply(lambda row:
math.sin(360/7*(row[col_index])),axis=1)
df["ctlog_latest_dow_cos"] = df.apply(lambda row:
math.cos(360/7*(row[col_index])),axis=1)


df.plot.scatter(x="ctlog_earliest_dow_sin",
y="ctlog_earliest_dow_cos").set_aspect('equal')
df.plot.scatter(x="whois_created_dow_sin",
y="whois_created_dow_cos").set_aspect('equal')
df.plot.scatter(x="ctlog_latest_dow_sin",
y="ctlog_latest_dow_cos").set_aspect('equal')

"""
# add one hot encode ctlog_earliest_not_before_dayofweek
df = pd.get_dummies(
        df,
        columns=["ctlog_earliest_dayofweek"],
        prefix=["ctlog_earliest_dow"],
    )
# add one hot encode whois_created_dayofweek to columns
df = pd.get_dummies(
        df, columns=["whois_created_dayofweek"], prefix="whois_dow"
    )
    """

# Summary statistics
click.echo(df.describe(include='all'))
                   domain malicious                  whois_created
count               21549     21549                          21549  \
unique              21536         2                            NaN
top     www.mediafire.com     False                            NaN
freq                    2     11739                            NaN
mean                  NaN       NaN  2012-10-03 12:56:32.335050496
min                   NaN       NaN            1986-01-09 00:00:00
```

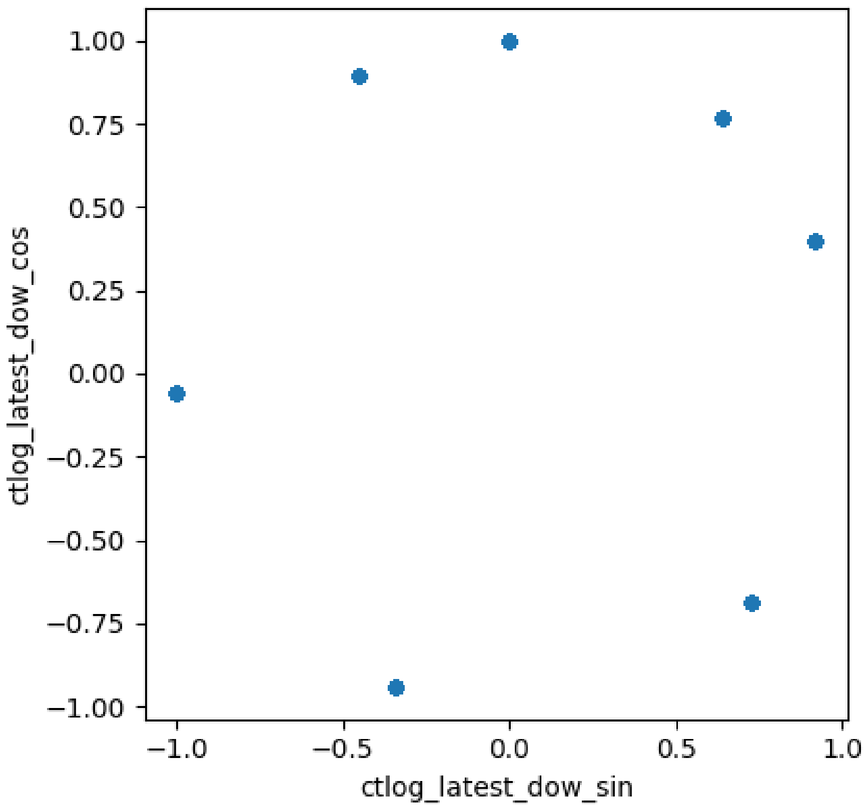
```
25%                    NaN       NaN             2003-05-25 13:35:05
50%                    NaN       NaN             2015-05-07 23:56:05
75%                    NaN       NaN             2023-03-20 15:03:16
max                    NaN       NaN             2023-07-03 08:21:24
std                    NaN       NaN                             NaN

                     ctlog_earliest                    ctlog_latest
count                         21549                           21549  \
unique                          NaN                             NaN
top                             NaN                             NaN
freq                            NaN                             NaN
mean    2022-09-26 15:45:50.943570432  2023-08-14 17:49:06.400900352
min             2021-11-30 05:24:28             2023-01-01 18:42:11
25%             2022-06-24 13:47:12             2023-07-02 08:11:07
50%             2022-10-18 21:00:14             2023-08-21 21:40:11
75%             2022-12-14 00:00:00             2023-09-21 19:41:38
max             2023-06-28 04:36:22             2023-12-31 23:59:59
std                             NaN                             NaN

       ctlog_wildcard  whois_created_dayofweek  ctlog_earliest_dayofweek
count           21549             21549.000000              21549.000000  \
unique              2                      NaN                       NaN
top             False                      NaN                       NaN
freq            13032                      NaN                       NaN
mean              NaN                 2.332823                  2.399462
min               NaN                 0.000000                  0.000000
25%               NaN                 1.000000                  1.000000
50%               NaN                 2.000000                  2.000000
75%               NaN                 4.000000                  4.000000
max               NaN                 6.000000                  6.000000
std               NaN                 1.775043                  1.897252

        ctlog_latest_dayofweek  domain_to_earliest_cert_delta
count             21549.000000                   21549.000000  \
unique                     NaN                            NaN
top                        NaN                            NaN
freq                       NaN                            NaN
mean                  2.873080                    3742.948397
min                   0.000000                       0.000000
25%                   1.000000                     181.000000
50%                   3.000000                    2637.000000
75%                   5.000000                    7078.000000
max                   6.000000                   13445.000000
std                   2.057394                    3694.584062

        domain_to_latest_cert_delta  whois_created_dow_sin
count                  21549.000000           21549.000000  \
unique                          NaN                    NaN
top                             NaN                    NaN
```

```
freq                          NaN                   NaN
mean                  3969.491206              0.140419
min                      0.000000             -0.998199
25%                    144.000000             -0.340712
50%                   3009.000000              0.000000
75%                   7421.000000              0.728010
max                  13798.000000              0.918032
std                   3850.835626              0.659922

        whois_created_dow_cos  ctlog_earliest_dow_sin
ctlog_earliest_dow_cos
count            21549.000000            21549.000000
21549.000000  \
unique                    NaN                     NaN
NaN
top                       NaN                     NaN
NaN
freq                      NaN                     NaN
NaN
mean                 0.054288                0.095357
0.161451
min                 -0.940168               -0.998199               -
0.940168
25%                 -0.685567               -0.340712               -
0.685567
50%                  0.396506                0.000000
0.396506
75%                  0.767830                0.728010
0.892589
max                  1.000000                0.918032
1.000000
std                  0.736128                0.651782
0.734891

        ctlog_latest_dow_sin  ctlog_latest_dow_cos
count           21549.000000          21549.000000
unique                   NaN                   NaN
top                      NaN                   NaN
freq                     NaN                   NaN
mean                0.096253              0.255578
min                -0.998199             -0.940168
25%                -0.450871             -0.685567
50%                 0.000000              0.396506
75%                 0.728010              0.892589
max                 0.918032              1.000000
std                 0.651597              0.707728
```


whois_created
ctlog_earliest
whois_created_dayofweek
ctlog_earliest_dayofweek
domain_to_earliest_cert_delta
domain_to_latest_cert_delta
1985 1990 1995 2000 2005 2010 2015 2020 2025
2022-01 2022-03 2022-05 2022-07 2022-09 2022-11 2023-01 2023-03 2023-05 2023-07
0 2500 5000 7500 10000 12500
0 5000 10000


1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
ctlog_earliest_dow_cos
−1.0
−0.5
0.0
0.5
1.0
ctlog_earliest_dow_sin

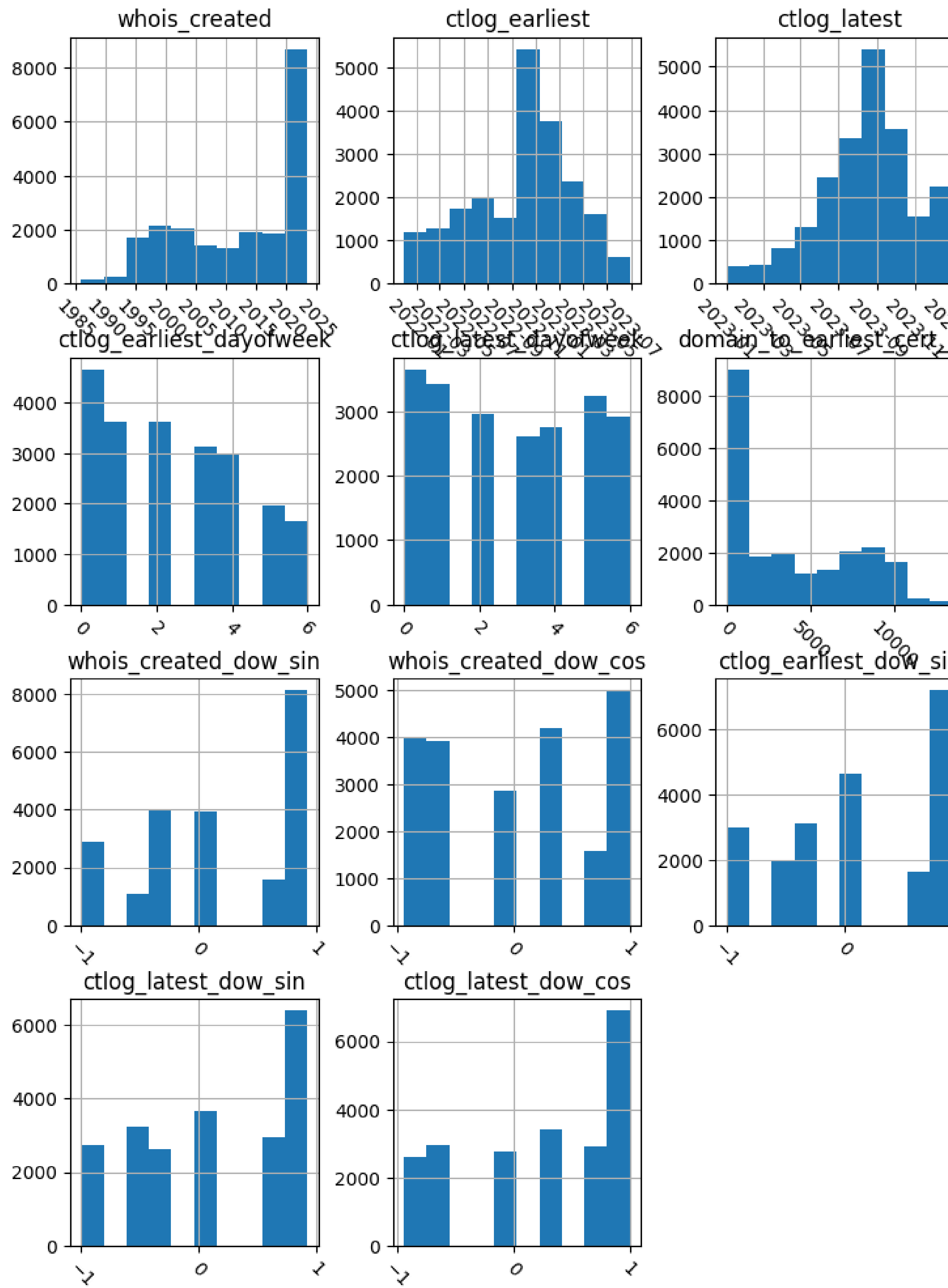

1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
whois_created_dow_cos
−1.0
−0.5
0.0
0.5
1.0
whois_created_dow_sin



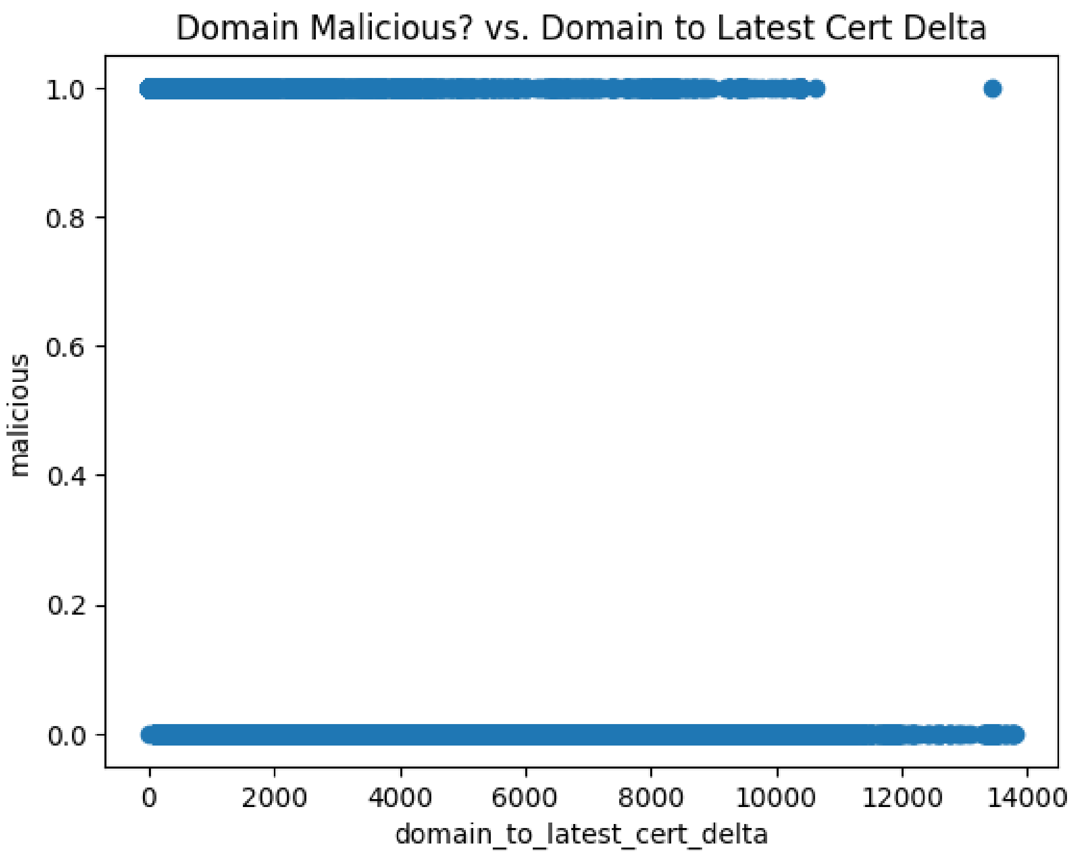
In [4]:

```
# Plot histograms
df.hist(figsize=(12,12), xrot=-45)

# Create a scatter plot of the data, showing malicious and benign domains
# plot the data as a scatter plot
plt.scatter(df["domain_to_earliest_cert_delta"], df["malicious"])
plt.xlabel("domain_to_earliest_cert_delta")
plt.ylabel("malicious")
plt.title("Domain Malicious? vs. Domain to Earliest Cert Delta")
plt.show()
# and for the latest
plt.scatter(df["domain_to_latest_cert_delta"], df["malicious"])
plt.xlabel("domain_to_latest_cert_delta")
plt.ylabel("malicious")
plt.title("Domain Malicious? vs. Domain to Latest Cert Delta")
plt.show()



click.echo(df.head())

# print a count of malicious and benign values
click.echo(df["malicious"].value_counts())
# deduplicate any rows, there shouldn't be any, but just in case
df = df.drop_duplicates()
click.echo(df["malicious"].value_counts())
```

```
click.echo(df.head())

X = df.drop(["malicious","domain", "ctlog_earliest", "ctlog_latest",
"whois_created", "whois_created_dayofweek", "ctlog_earliest_dayofweek"],
axis=1)
X = df.filter(combo_features)
y = df.filter(["malicious"])


click.echo(X.describe())
```



whois_created
ctlog_earliest
ctlog_latest
ctlog_earliest_dayofweek
ctlog_latest_dayofweek
domain_to_earliest_cert_
whois_created_dow_sin
whois_created_dow_cos
ctlog_earliest_dow_si
ctlog_latest_dow_sin
ctlog_latest_dow_cos




```
                              domain  malicious        whois_created  
0  i-db5p-cor001.api.p001.1drv.com      False 2013-08-05 18:33:50   \
4        soundcloud-pax.pandora.com      False 1993-12-28 05:00:00   
5          joolcomercializadora.com       True 2023-05-22 14:53:50   
6         createpdf-asr.acrobat.com      False 1999-03-16 05:00:00   
8                           popt.in      False 2016-05-14 16:58:55   

        ctlog_earliest        ctlog_latest  ctlog_wildcard  
0 2022-01-24 20:01:58 2023-06-08 20:46:06            True   \
4 2022-05-19 00:00:00 2023-06-19 23:59:59            True   
5 2022-04-06 22:23:24 2023-09-22 23:59:59           False   
6 2022-09-09 00:00:00 2023-10-10 23:59:59            True   
8 2023-01-07 20:36:15 2023-08-15 04:16:52           False   

   whois_created_dayofweek  ctlog_earliest_dayofweek  
ctlog_latest_dayofweek  
0                        0                          0  
3   \
4                        1                          3  
0  
5                        0                          2  
4  
6                        1                          4  
1  
8                        5                          5  
1  
```

```
   domain_to_earliest_cert_delta  domain_to_latest_cert_delta
0                         3095.0                       3595.0  \
4                        10369.0                      10766.0
5                          410.0                        124.0
6                         8578.0                       8975.0
8                         2430.0                       2649.0

   whois_created_dow_sin  whois_created_dow_cos  ctlog_earliest_dow_sin
0               0.000000               1.000000                0.000000  \
4               0.918032               0.396506               -0.340712
5               0.000000               1.000000                0.728010
6               0.918032               0.396506               -0.998199
8              -0.450871               0.892589               -0.450871

   ctlog_earliest_dow_cos  ctlog_latest_dow_sin  ctlog_latest_dow_cos
0                1.000000             -0.340712             -0.940168
4               -0.940168              0.000000              1.000000
5               -0.685567             -0.998199             -0.059997
6               -0.059997              0.918032              0.396506
8                0.892589              0.918032              0.396506
malicious
False    11739
True      9810
Name: count, dtype: int64
malicious
False    11739
True      9810
Name: count, dtype: int64
                            domain  malicious       whois_created
0  i-db5p-cor001.api.p001.1drv.com      False 2013-08-05 18:33:50  \
4       soundcloud-pax.pandora.com      False 1993-12-28 05:00:00
5         joolcomercializadora.com       True 2023-05-22 14:53:50
6        createpdf-asr.acrobat.com      False 1999-03-16 05:00:00
8                          popt.in      False 2016-05-14 16:58:55

       ctlog_earliest        ctlog_latest  ctlog_wildcard
0 2022-01-24 20:01:58 2023-06-08 20:46:06            True  \
4 2022-05-19 00:00:00 2023-06-19 23:59:59            True
5 2022-04-06 22:23:24 2023-09-22 23:59:59           False
6 2022-09-09 00:00:00 2023-10-10 23:59:59            True
8 2023-01-07 20:36:15 2023-08-15 04:16:52           False

   whois_created_dayofweek  ctlog_earliest_dayofweek
ctlog_latest_dayofweek
0                        0                         0
3  \
4                        1                         3
0
```

```
5                    0                          2
4
6                    1                          4
1
8                    5                          5
1

   domain_to_earliest_cert_delta  domain_to_latest_cert_delta
0                         3095.0                       3595.0  \
4                        10369.0                      10766.0
5                          410.0                        124.0
6                         8578.0                       8975.0
8                         2430.0                       2649.0

   whois_created_dow_sin  whois_created_dow_cos  ctlog_earliest_dow_sin
0               0.000000               1.000000                0.000000  \
4               0.918032               0.396506               -0.340712
5               0.000000               1.000000                0.728010
6               0.918032               0.396506               -0.998199
8              -0.450871               0.892589               -0.450871

   ctlog_earliest_dow_cos  ctlog_latest_dow_sin  ctlog_latest_dow_cos
0                1.000000             -0.340712             -0.940168
4               -0.940168              0.000000              1.000000
5               -0.685567             -0.998199             -0.059997
6               -0.059997              0.918032              0.396506
8                0.892589              0.918032              0.396506
       domain_to_earliest_cert_delta  ctlog_earliest_dow_sin
count                   21549.000000            21549.000000  \
mean                     3742.948397                0.095357
std                      3694.584062                0.651782
min                         0.000000               -0.998199
25%                       181.000000               -0.340712
50%                      2637.000000                0.000000
75%                      7078.000000                0.728010
max                     13445.000000                0.918032

       ctlog_earliest_dow_cos
count            21549.000000
mean                 0.161451
std                  0.734891
min                 -0.940168
25%                 -0.685567
50%                  0.396506
75%                  0.892589
max                  1.000000
```

In [5]:

```
# convert y (malicious) to 1/0 int
y = y.astype('int')
```

```
if "ctlog_wildcard" in X.columns:
    X["ctlog_wildcard"] = X["ctlog_wildcard"].astype('int')

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)


# random forest model

param_grid = {
    'n_estimators': [50,100,150,200],
    'max_features': ['sqrt', 'log2'],
    'max_depth' : [2,3,4,5],
    'criterion' :['gini', 'entropy']
}
```

In [6]:

```
rf = RandomForestClassifier(random_state=42)
rf_cv = GridSearchCV(estimator=rf, param_grid=param_grid, cv=5, n_jobs=-1)
rf_cv.fit(X_train, y_train.values.ravel())
```

Out[6]:

GridSearchCV

```
GridSearchCV(cv=5, estimator=RandomForestClassifier(random_state=42),
n_jobs=-1,
             param_grid={'criterion': ['gini', 'entropy'],
                         'max_depth': [2, 3, 4, 5],
                         'max_features': ['sqrt', 'log2'],
                         'n_estimators': [50, 100, 150, 200]})
```

estimator: RandomForestClassifier

```
RandomForestClassifier(random_state=42)
```

```
RandomForestClassifier
RandomForestClassifier(random_state=42)
```

In [7]:

```
bp = rf_cv.best_params_
click.echo("Best parameters set found:")
click.echo(bp)
Best parameters set found:
{'criterion': 'gini', 'max_depth': 5, 'max_features': 'sqrt',
'n_estimators': 150}
```

In [8]:

```
rf = RandomForestClassifier(random_state=42,
max_features=bp["max_features"], n_estimators=bp["n_estimators"],
max_depth=bp["max_depth"], criterion=bp["criterion"])
```

In [9]:

```
rf.fit(X_train, y_train.values.ravel())
```

Out[9]:

```
RandomForestClassifier
RandomForestClassifier(max_depth=5, n_estimators=150, random_state=42)
```

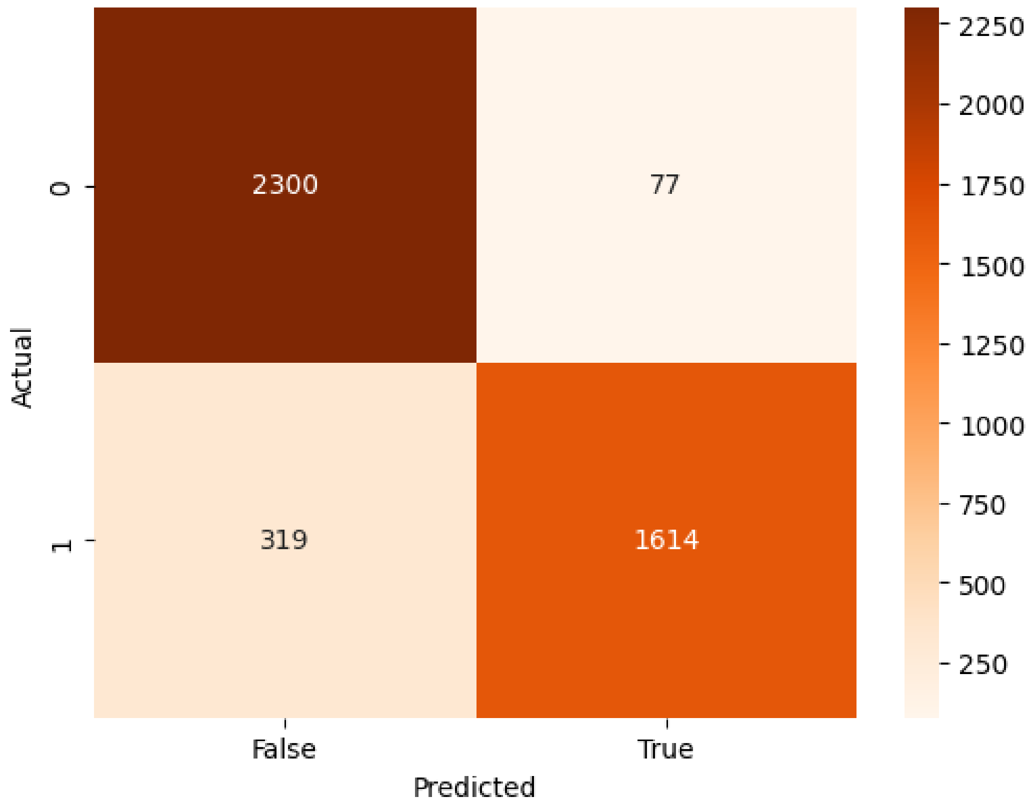
In [ ]:

In [10]:

```
# Predict the malicious column using the test data
#add the incepts


y_predicted = rf.predict(X_test)

# Present the results
click.echo("Features selected:")
click.echo(X.columns)
click.echo("Confusion matrix:")
cm = confusion_matrix(y_test, y_predicted)
click.echo(cm)
click.echo("Classification report:")
click.echo(classification_report(y_test, y_predicted))


# Heatmap of confusion matrix
y_predicted

threshold = 0.5
y_predicted = [y > threshold for y in y_predicted]
data = {'Actual':    y_test.values.flatten(),
        'Predicted': y_predicted
        }

# Generate a confusion matrix and heatmap to evaluate the Type I and Type
II errors/ FP/FN etc.
df = pd.DataFrame(data, columns=['Actual','Predicted'])
cm2 = pd.crosstab(df['Actual'], df['Predicted'], rownames=['Actual'],
colnames=['Predicted'])
fig = sns.heatmap(cm2, annot=True, cmap='Oranges', fmt='g')
fig
Features selected:
Index(['domain_to_earliest_cert_delta', 'ctlog_earliest_dow_sin',
       'ctlog_earliest_dow_cos', 'ctlog_wildcard'],
      dtype='object')
Confusion matrix:
[[2300   77]
 [ 319 1614]]
Classification report:
              precision    recall  f1-score   support

           0       0.88      0.97      0.92      2377
           1       0.95      0.83      0.89      1933

    accuracy                           0.91      4310
   macro avg       0.92      0.90      0.91      4310
weighted avg       0.91      0.91      0.91      4310
```

Out[10]:

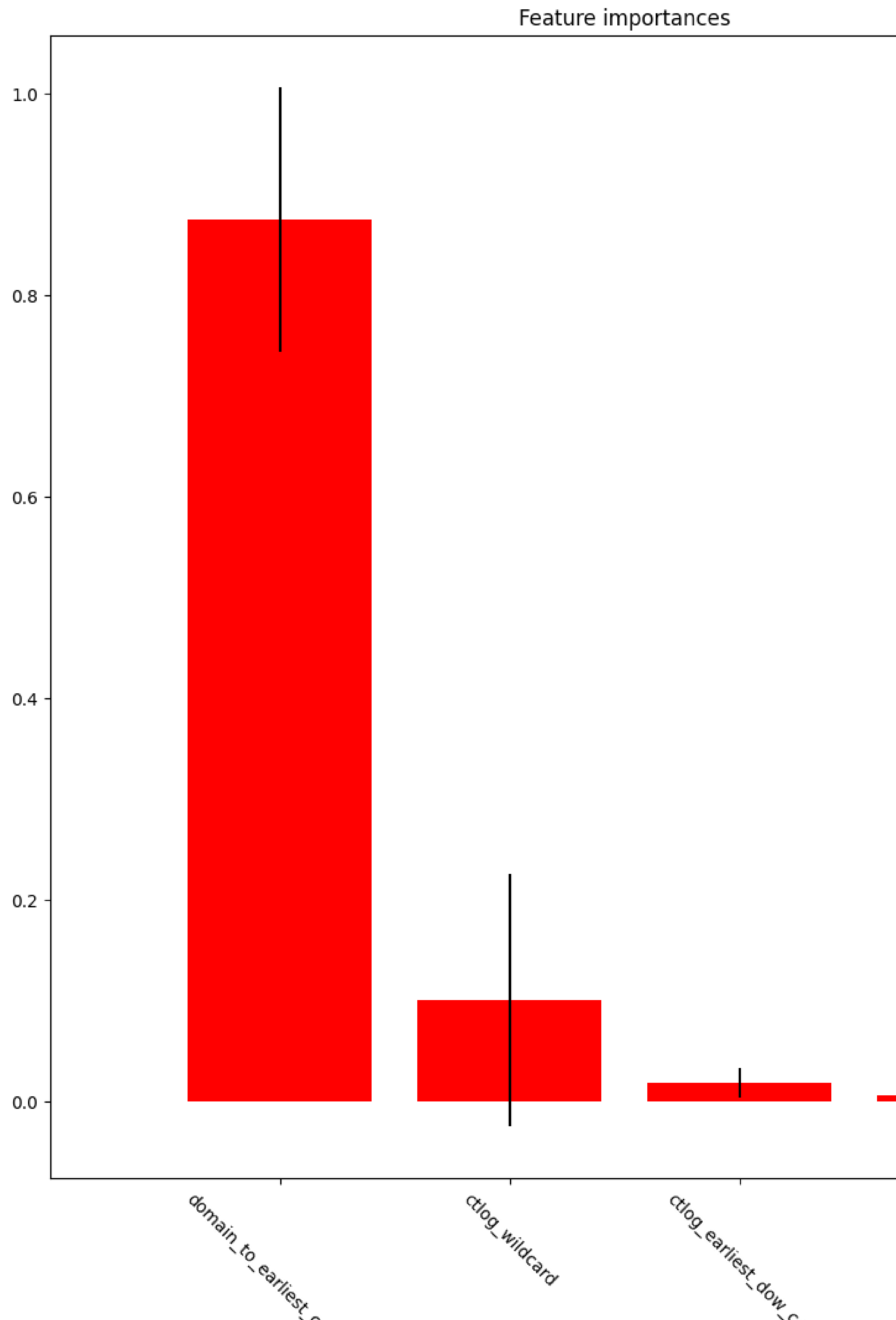
```
<Axes: xlabel='Predicted', ylabel='Actual'>
```



In [11]:

```
# plot the feature importances
importances = rf.feature_importances_
std = np.std([tree.feature_importances_ for tree in rf.estimators_],
axis=0)

indices = np.argsort(importances)[::-1]
# Print the feature ranking
click.echo("Feature ranking:")
for f in range(X.shape[1]):
    click.echo("%d. feature %s (%f)" % (f + 1, combo_features[indices[f]],
importances[indices[f]]))

# Plot the feature importances of the forest
plt.figure(figsize=(12,12))
plt.title("Feature importances")
plt.bar(range(X.shape[1]), importances[indices], color="r",
yerr=std[indices], align="center")
plt.xticks(range(X.shape[1]), X.columns[indices], rotation=-45)
plt.xlim([-1, X.shape[1]])
plt.show()
Feature ranking:
1. feature domain_to_earliest_cert_delta (0.874685)
2. feature ctlog_wildcard (0.101094)
```

```
3. feature ctlog_earliest_dow_cos (0.018101)
4. feature ctlog_earliest_dow_sin (0.006121)
```


Feature importances
1.0
0.8
0.6
0.4
0.2
0.0
ctlog_wildcard

V. Feature Set D

In [1]:

```
import click
import pandas as pd
import glob
import os
#import sklearn
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.metrics import accuracy_score, f1_score, confusion_matrix,
classification_report, roc_auc_score, roc_curve, auc
from sklearn.preprocessing import StandardScaler
from scipy import stats
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
import statsmodels.api as sm
import matplotlib.pyplot as plt
from datetime import datetime, date
# qqplot of the data
import seaborn as sns
import math
import numpy as np
import utils

combo_features =     ['domain_to_earliest_cert_delta',
'domain_to_latest_cert_delta']

path = "../data/"
filename = None

epoch = datetime(1970,1,1)
# open the training data file, from ../data/ for the most recent merged
training data file saved by prepare-training-data-parallel.py
full_path = path + "merged_20230705-104357_training_adorned-engineered.csv"

# get latest file matching the glob
if not filename:
    filename = max(glob.glob(full_path), key=os.path.getctime)
    click.echo(f"Using {filename} as training data")
    df = pd.read_csv(filename, sep=",")

# randomize the rows
df = df.sample(frac=1, random_state=42).reset_index(drop=True)

click.echo(df.shape)
click.echo(df.columns)

""" # From prepare-training-data-parallel.py:
# Columns:
url
```

```
verification_time
domain
malicious
dateadded
whois_created
soa_timestamp
ctlog_earliest
ctlog_latest
ctlog_wildcard
whois_created_dayofweek,
ctlog_earliest_dayofweek,
domain_to_cert_delta
"""


# Data cleaning - there may be some epoch values in the whois_created
#  & ct_log_earliest columns, so we need to remove those rows

# convert the whois_created and ctlog_earliest, ctlog_latest columns to
datetime

df = df.loc[df["whois_created"] != ""]
df = df.loc[df["ctlog_earliest"] != ""]
df = df.loc[df["ctlog_latest"] != ""]

# some dates are have nanoseconds in them, so we need to remove the
nanosecond part
df["whois_created"] = df["whois_created"].str.split(".", n = 1, expand =
True)[0]
# some dates has additional timezone information, so we need to remove that
df["whois_created"] = df["whois_created"].apply(lambda x: " ".join(
str(x).split(" ")[:2]))


# convert the whois_created and ct_log_earliest columns to datetime
df = df.astype({"whois_created": 'datetime64[ns]', "ctlog_earliest":
'datetime64[ns]', "ctlog_latest": 'datetime64[ns]'})
df = df.loc[df["whois_created"].ne(pd.Timestamp('1970-01-01 00:00:00'))]
df = df.loc[df["ctlog_earliest"].ne(pd.Timestamp('1970-01-01 00:00:00'))]
df = df.loc[df["ctlog_latest"].ne(pd.Timestamp('1970-01-01 00:00:00'))]


# malicious is a bool
df['malicious'] = df['malicious'].astype('bool')
df['domain'] = df['domain'].astype('string')
Using ../data/merged_20230705-104357_training_adorned-engineered.csv as
training data
(35438, 13)
Index(['url', 'verification_time', 'domain', 'malicious', 'dateadded',
```

```
       'whois_created', 'soa_timestamp', 'ctlog_earliest', 'ctlog_latest',
       'ctlog_wildcard', 'whois_created_dayofweek',
'ctlog_earliest_dayofweek',
       'domain_to_cert_delta'],
      dtype='object')
```

In [2]:

```
#########
# Feature engineering
#########

# remove any rows where the whois_created or ctlog_earliest columns are
null
df = df.loc[df["whois_created"].notnull()]
df = df.loc[df["ctlog_earliest"].notnull()]

# if the data is missing the "whois_created_dayofweek" or
"ctlog_earliest_dayofweek" columns, then add them
# whois created day of the week 0 = Monday, 6 = Sunday
df["whois_created_dayofweek"] = df["whois_created"].apply(
        lambda x: x.weekday() if isinstance(x, date) else None
    )

# cert valid day of the week 0 = Monday, 6 = Sunday
df["ctlog_earliest_dayofweek"] = df["ctlog_earliest"].apply(
        lambda x: x.weekday() if isinstance(x, date) else None
    )

df["ctlog_latest_dayofweek"] = df["ctlog_latest"].apply(
        lambda x: x.weekday() if isinstance(x, date) else None
    )

# set data type of the day of week columns to int
df["whois_created_dayofweek"] =
df["whois_created_dayofweek"].astype("int64")
df["ctlog_earliest_dayofweek"] =
df["ctlog_earliest_dayofweek"].astype("int64")
df["ctlog_latest_dayofweek"] = df["ctlog_latest_dayofweek"].astype("int64")

# domain to cert delta
df["domain_to_earliest_cert_delta"] = df.apply(
        lambda x: utils.get_days_delta(
            x["ctlog_earliest"],
            x["whois_created"]
            if x["whois_created"] and x["ctlog_earliest"]
            else None,
        ),
        axis=1,
    )
```

```
# set the data type of the domain_to_cert_delta column to float
df["domain_to_earliest_cert_delta"] =
df["domain_to_earliest_cert_delta"].astype("float64")

# domain to latest cert delta
df["domain_to_latest_cert_delta"] = df.apply(
        lambda x: utils.get_days_delta(
            x["ctlog_latest"],
            x["whois_created"]
            if x["whois_created"] and x["ctlog_latest"]
            else None,
        ),
        axis=1,
    )

# set the data type of the domain_to_cert_delta column to float
df["domain_to_latest_cert_delta"] =
df["domain_to_latest_cert_delta"].astype("float64")

# drop columns we imported that we will never use
df = df.drop(columns=["url", "verification_time", "dateadded",
"soa_timestamp","domain_to_cert_delta"])

click.echo(df.head())
click.echo(df.describe(include='all'))
click.echo(df.dtypes)
                            domain  malicious       whois_created
0  i-db5p-cor001.api.p001.1drv.com      False 2013-08-05 18:33:50  \
4       soundcloud-pax.pandora.com      False 1993-12-28 05:00:00
5         joolcomercializadora.com       True 2023-05-22 14:53:50
6        createpdf-asr.acrobat.com      False 1999-03-16 05:00:00
8                          popt.in      False 2016-05-14 16:58:55

       ctlog_earliest        ctlog_latest  ctlog_wildcard
0 2022-01-24 20:01:58 2023-06-08 20:46:06            True  \
4 2022-05-19 00:00:00 2023-06-19 23:59:59            True
5 2022-04-06 22:23:24 2023-09-22 23:59:59           False
6 2022-09-09 00:00:00 2023-10-10 23:59:59            True
8 2023-01-07 20:36:15 2023-08-15 04:16:52           False

   whois_created_dayofweek  ctlog_earliest_dayofweek
ctlog_latest_dayofweek
0                        0                          0
3  \
4                        1                          3
0
5                        0                          2
4
```

```
6                        1                          4
1
8                        5                          5
1

   domain_to_earliest_cert_delta  domain_to_latest_cert_delta
0                        -3095.0                      -3595.0
4                       -10369.0                     -10766.0
5                          410.0                       -124.0
6                        -8578.0                      -8975.0
8                        -2430.0                      -2649.0
                   domain malicious                  whois_created
count               21549     21549                          21549  \
unique              21536         2                            NaN
top      www.mediafire.com     False                            NaN
freq                    2     11739                            NaN
mean                  NaN       NaN  2012-10-03 12:56:32.335050496
min                   NaN       NaN            1986-01-09 00:00:00
25%                   NaN       NaN            2003-05-25 13:35:05
50%                   NaN       NaN            2015-05-07 23:56:05
75%                   NaN       NaN            2023-03-20 15:03:16
max                   NaN       NaN            2023-07-03 08:21:24
std                   NaN       NaN                            NaN

                       ctlog_earliest                   ctlog_latest
count                           21549                          21549  \
unique                            NaN                            NaN
top                               NaN                            NaN
freq                              NaN                            NaN
mean    2022-09-26 15:45:50.943570432  2023-08-14 17:49:06.400900352
min               2021-11-30 05:24:28            2023-01-01 18:42:11
25%               2022-06-24 13:47:12            2023-07-02 08:11:07
50%               2022-10-18 21:00:14            2023-08-21 21:40:11
75%               2022-12-14 00:00:00            2023-09-21 19:41:38
max               2023-06-28 04:36:22            2023-12-31 23:59:59
std                               NaN                            NaN

       ctlog_wildcard  whois_created_dayofweek  ctlog_earliest_dayofweek
count           21549             21549.000000              21549.000000  \
unique              2                      NaN                       NaN
top             False                      NaN                       NaN
freq            13032                      NaN                       NaN
mean              NaN                 2.332823                  2.399462
min               NaN                 0.000000                  0.000000
25%               NaN                 1.000000                  1.000000
50%               NaN                 2.000000                  2.000000
75%               NaN                 4.000000                  4.000000
max               NaN                 6.000000                  6.000000
std               NaN                 1.775043                  1.897252
```

```
       ctlog_latest_dayofweek  domain_to_earliest_cert_delta
count              21549.000000                   21549.000000  \
unique                      NaN                            NaN
top                         NaN                            NaN
freq                        NaN                            NaN
mean                   2.873080                   -3645.602070
min                    0.000000                  -13445.000000
25%                    1.000000                   -7078.000000
50%                    3.000000                   -2637.000000
75%                    5.000000                      69.000000
max                    6.000000                     524.000000
std                    2.057394                    3790.677119

       domain_to_latest_cert_delta
count                 21549.000000
unique                         NaN
top                            NaN
freq                           NaN
mean                  -3967.678222
min                  -13798.000000
25%                   -7421.000000
50%                   -3009.000000
75%                    -144.000000
max                     135.000000
std                    3852.703681
domain                              string[python]
malicious                                     bool
whois_created                       datetime64[ns]
ctlog_earliest                      datetime64[ns]
ctlog_latest                        datetime64[ns]
ctlog_wildcard                                bool
whois_created_dayofweek                      int64
ctlog_earliest_dayofweek                     int64
ctlog_latest_dayofweek                       int64
domain_to_earliest_cert_delta              float64
domain_to_latest_cert_delta                float64
dtype: object
```

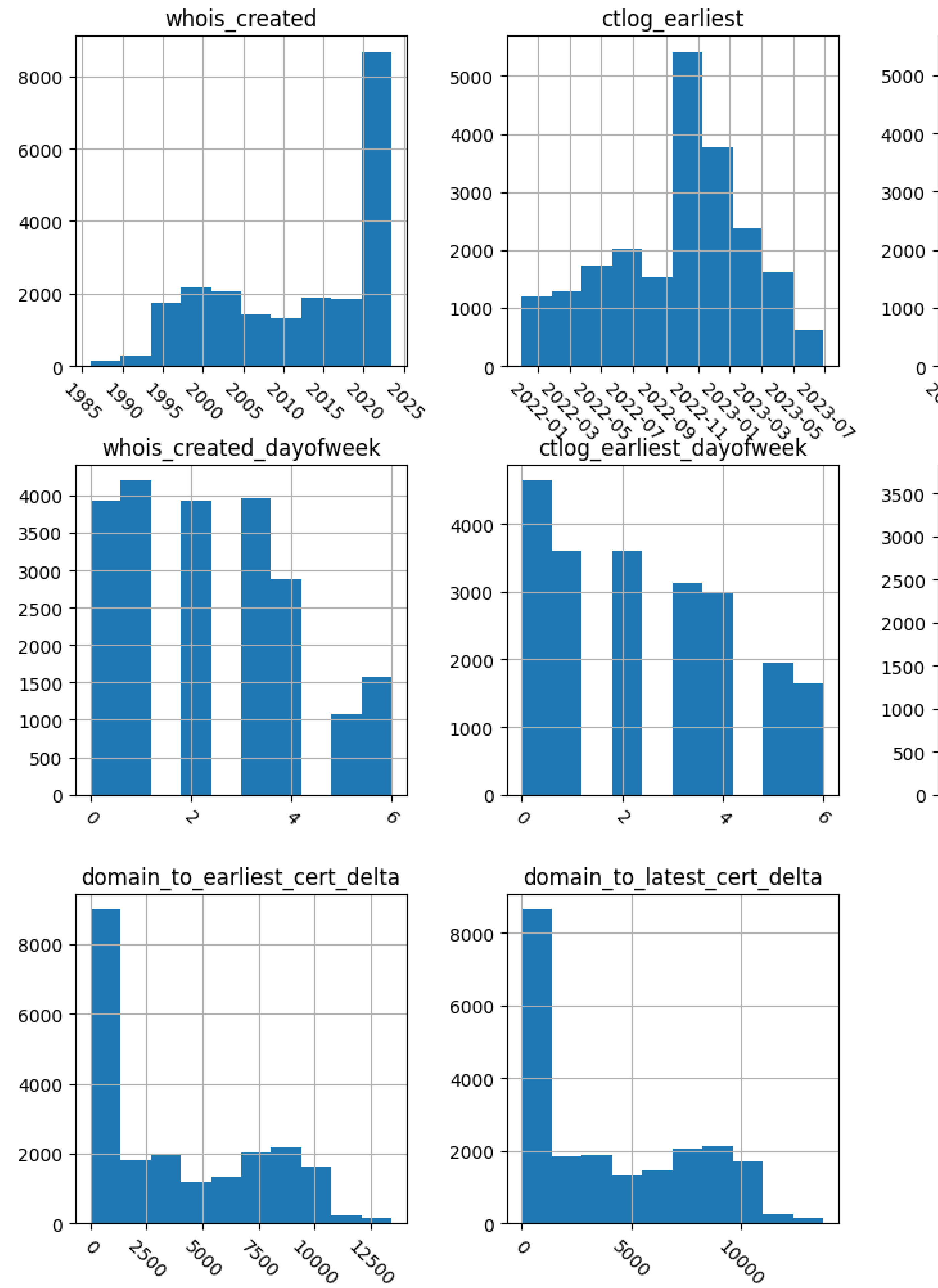
In [3]:

```
# absolute value of the domain_to_cert_delta
df["domain_to_earliest_cert_delta"] =
abs(df["domain_to_earliest_cert_delta"])
df["domain_to_latest_cert_delta"] = abs(df["domain_to_latest_cert_delta"])

df.hist(figsize=(12,12), xrot=-45)

col_index = df.columns.get_loc("whois_created_dayofweek")
df["whois_created_dow_sin"] = df.apply(lambda row:
math.sin(360/7*(row[col_index])),axis=1)
```

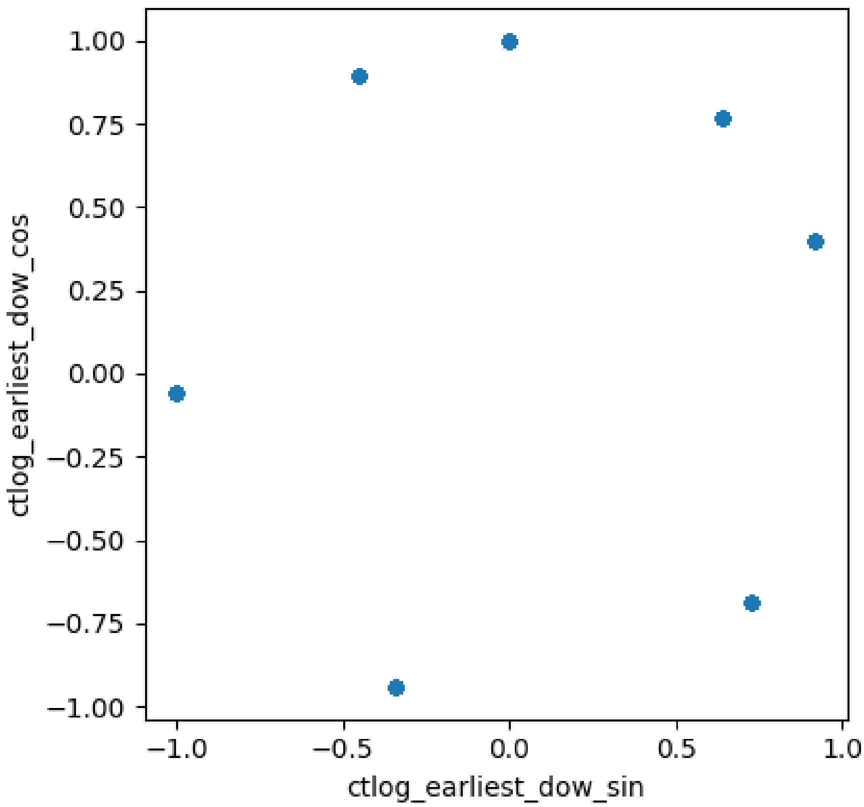
```
df["whois_created_dow_cos"] = df.apply(lambda row:
math.cos(360/7*(row[col_index])),axis=1)

col_index = df.columns.get_loc("ctlog_earliest_dayofweek")
df["ctlog_earliest_dow_sin"] = df.apply(lambda row:
math.sin(360/7*(row[col_index])),axis=1)
df["ctlog_earliest_dow_cos"] = df.apply(lambda row:
math.cos(360/7*(row[col_index])),axis=1)

col_index = df.columns.get_loc("ctlog_latest_dayofweek")
df["ctlog_latest_dow_sin"] = df.apply(lambda row:
math.sin(360/7*(row[col_index])),axis=1)
df["ctlog_latest_dow_cos"] = df.apply(lambda row:
math.cos(360/7*(row[col_index])),axis=1)


df.plot.scatter(x="ctlog_earliest_dow_sin",
y="ctlog_earliest_dow_cos").set_aspect('equal')
df.plot.scatter(x="whois_created_dow_sin",
y="whois_created_dow_cos").set_aspect('equal')
df.plot.scatter(x="ctlog_latest_dow_sin",
y="ctlog_latest_dow_cos").set_aspect('equal')

"""
# add one hot encode ctlog_earliest_not_before_dayofweek
df = pd.get_dummies(
        df,
        columns=["ctlog_earliest_dayofweek"],
        prefix=["ctlog_earliest_dow"],
    )
# add one hot encode whois_created_dayofweek to columns
df = pd.get_dummies(
        df, columns=["whois_created_dayofweek"], prefix="whois_dow"
    )
    """

# Summary statistics
click.echo(df.describe(include='all'))
                   domain malicious                  whois_created
count               21549     21549                          21549  \
unique              21536         2                            NaN
top     www.mediafire.com     False                            NaN
freq                    2     11739                            NaN
mean                  NaN       NaN  2012-10-03 12:56:32.335050496
min                   NaN       NaN            1986-01-09 00:00:00
25%                   NaN       NaN            2003-05-25 13:35:05
50%                   NaN       NaN            2015-05-07 23:56:05
75%                   NaN       NaN            2023-03-20 15:03:16
max                   NaN       NaN            2023-07-03 08:21:24
```

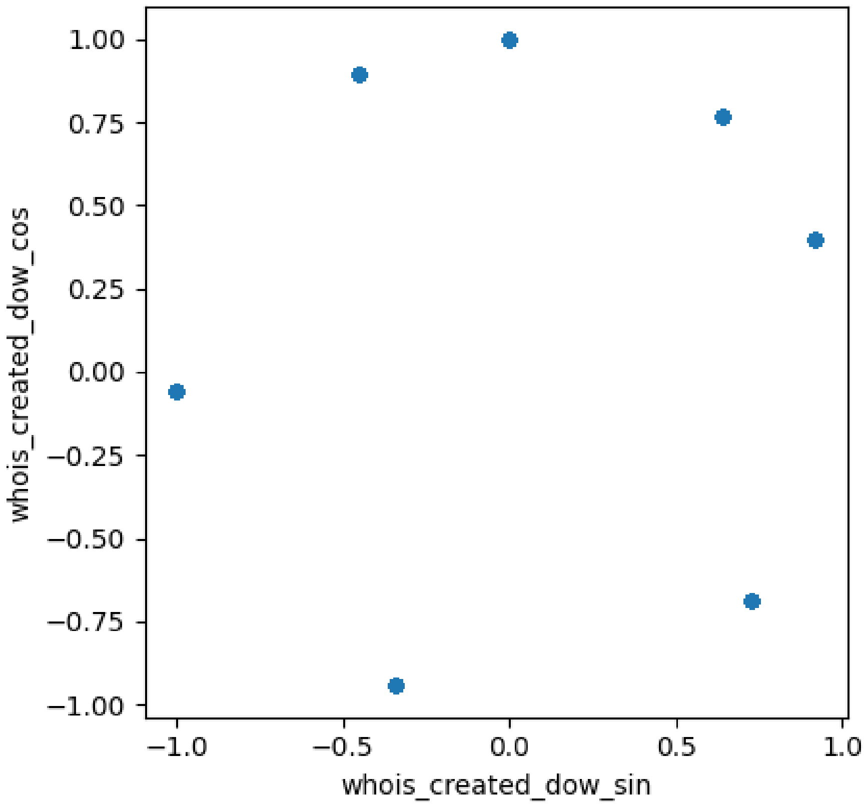
```
std                   NaN       NaN                             NaN

                      ctlog_earliest                 ctlog_latest
count                          21549                        21549  \
unique                           NaN                          NaN
top                              NaN                          NaN
freq                             NaN                          NaN
mean    2022-09-26 15:45:50.943570432  2023-08-14 17:49:06.400900352
min              2021-11-30 05:24:28          2023-01-01 18:42:11
25%              2022-06-24 13:47:12          2023-07-02 08:11:07
50%              2022-10-18 21:00:14          2023-08-21 21:40:11
75%              2022-12-14 00:00:00          2023-09-21 19:41:38
max              2023-06-28 04:36:22          2023-12-31 23:59:59
std                              NaN                          NaN

       ctlog_wildcard  whois_created_dayofweek  ctlog_earliest_dayofweek
count           21549             21549.000000              21549.000000  \
unique              2                      NaN                       NaN
top             False                      NaN                       NaN
freq            13032                      NaN                       NaN
mean              NaN                 2.332823                  2.399462
min               NaN                 0.000000                  0.000000
25%               NaN                 1.000000                  1.000000
50%               NaN                 2.000000                  2.000000
75%               NaN                 4.000000                  4.000000
max               NaN                 6.000000                  6.000000
std               NaN                 1.775043                  1.897252

        ctlog_latest_dayofweek  domain_to_earliest_cert_delta
count             21549.000000                   21549.000000  \
unique                     NaN                            NaN
top                        NaN                            NaN
freq                       NaN                            NaN
mean                  2.873080                    3742.948397
min                   0.000000                       0.000000
25%                   1.000000                     181.000000
50%                   3.000000                    2637.000000
75%                   5.000000                    7078.000000
max                   6.000000                   13445.000000
std                   2.057394                    3694.584062

        domain_to_latest_cert_delta  whois_created_dow_sin
count                  21549.000000           21549.000000  \
unique                          NaN                    NaN
top                             NaN                    NaN
freq                            NaN                    NaN
mean                    3969.491206               0.140419
min                        0.000000              -0.998199
25%                      144.000000              -0.340712
```

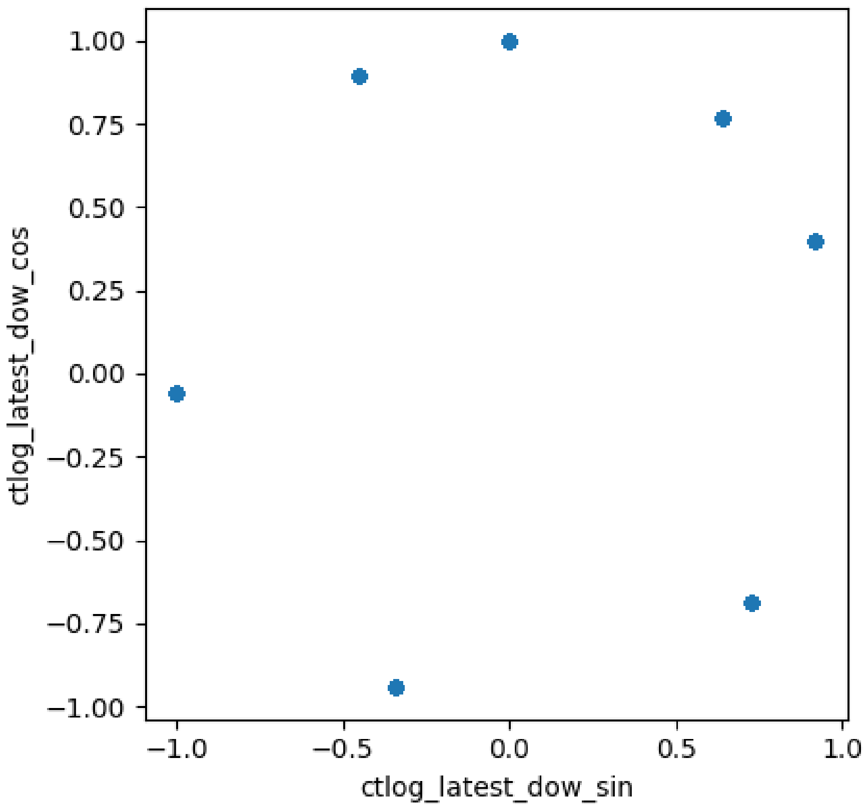
```
50%                    3009.000000              0.000000
75%                    7421.000000              0.728010
max                   13798.000000              0.918032
std                    3850.835626              0.659922

        whois_created_dow_cos  ctlog_earliest_dow_sin
ctlog_earliest_dow_cos
count            21549.000000            21549.000000
21549.000000  \
unique                    NaN                     NaN
NaN
top                       NaN                     NaN
NaN
freq                      NaN                     NaN
NaN
mean                 0.054288                0.095357
0.161451
min                 -0.940168               -0.998199               -
0.940168
25%                 -0.685567               -0.340712               -
0.685567
50%                  0.396506                0.000000
0.396506
75%                  0.767830                0.728010
0.892589
max                  1.000000                0.918032
1.000000
std                  0.736128                0.651782
0.734891

        ctlog_latest_dow_sin  ctlog_latest_dow_cos
count           21549.000000          21549.000000
unique                   NaN                   NaN
top                      NaN                   NaN
freq                     NaN                   NaN
mean                0.096253              0.255578
min                -0.998199             -0.940168
25%                -0.450871             -0.685567
50%                 0.000000              0.396506
75%                 0.728010              0.892589
max                 0.918032              1.000000
std                 0.651597              0.707728
```



whois_created
0
2000
4000
6000
8000
1985
1990
1995
2000
2005
2010
2015
2020
2025
ctlog_earliest
0
1000
2000
3000
4000
5000
2022-01
2022-03
2022-05
2022-07
2022-09
2022-11
2023-01
2023-03
2023-05
2023-07
whois_created_dayofweek
0
500
1000
1500
2000
2500
3000
3500
4000
0
2
4
6
ctlog_earliest_dayofweek
0
1000
2000
3000
4000
0
2
4
6
domain_to_earliest_cert_delta
0
2000
4000
6000
8000
0
2500
5000
7500
10000
12500
domain_to_latest_cert_delta
0
2000
4000
6000
8000
0
5000
10000


1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
ctlog_earliest_dow_cos
−1.0
−0.5
0.0
0.5
1.0
ctlog_earliest_dow_sin

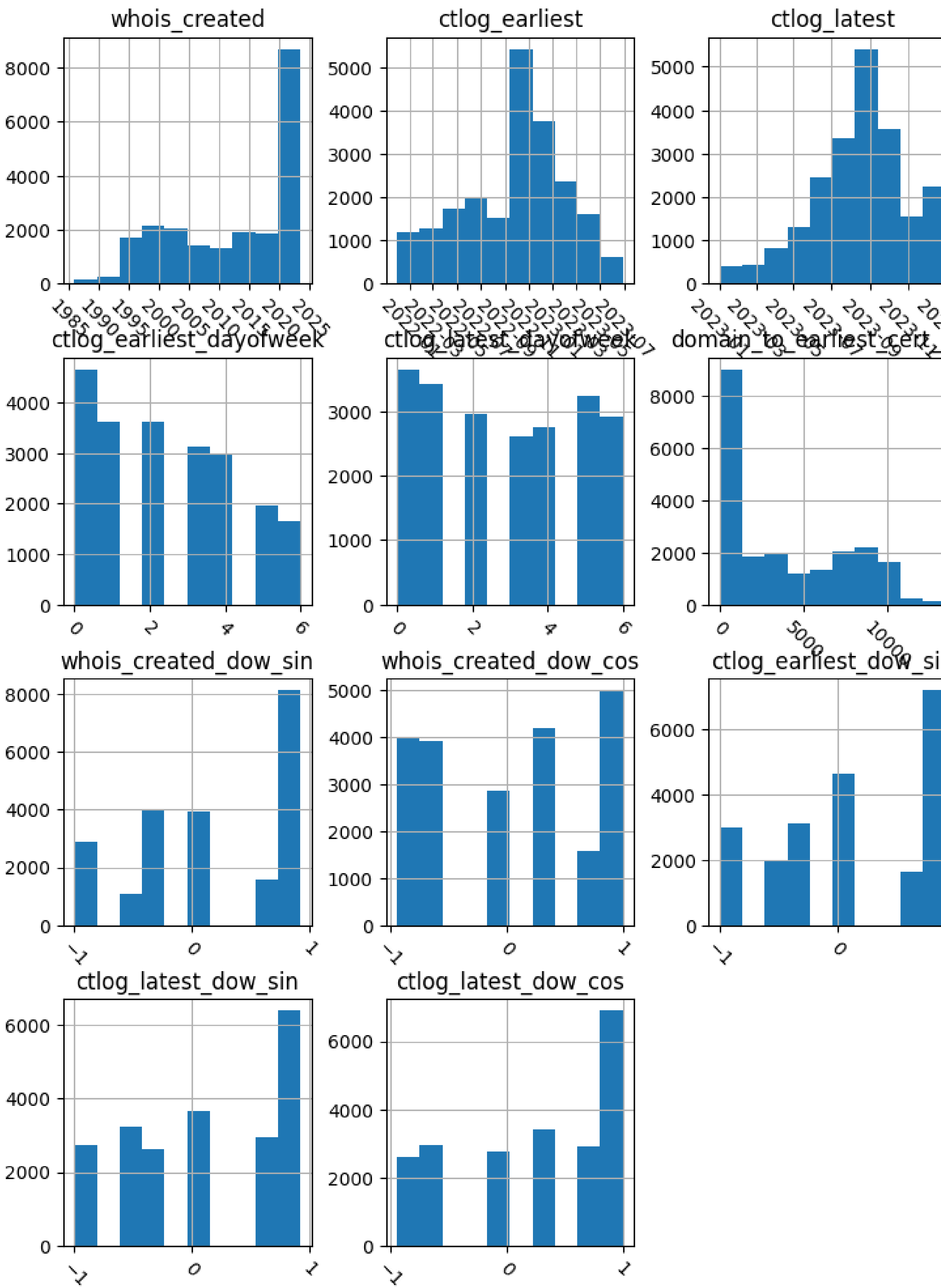

1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
whois_created_dow_cos
−1.0
−0.5
0.0
0.5
1.0
whois_created_dow_sin

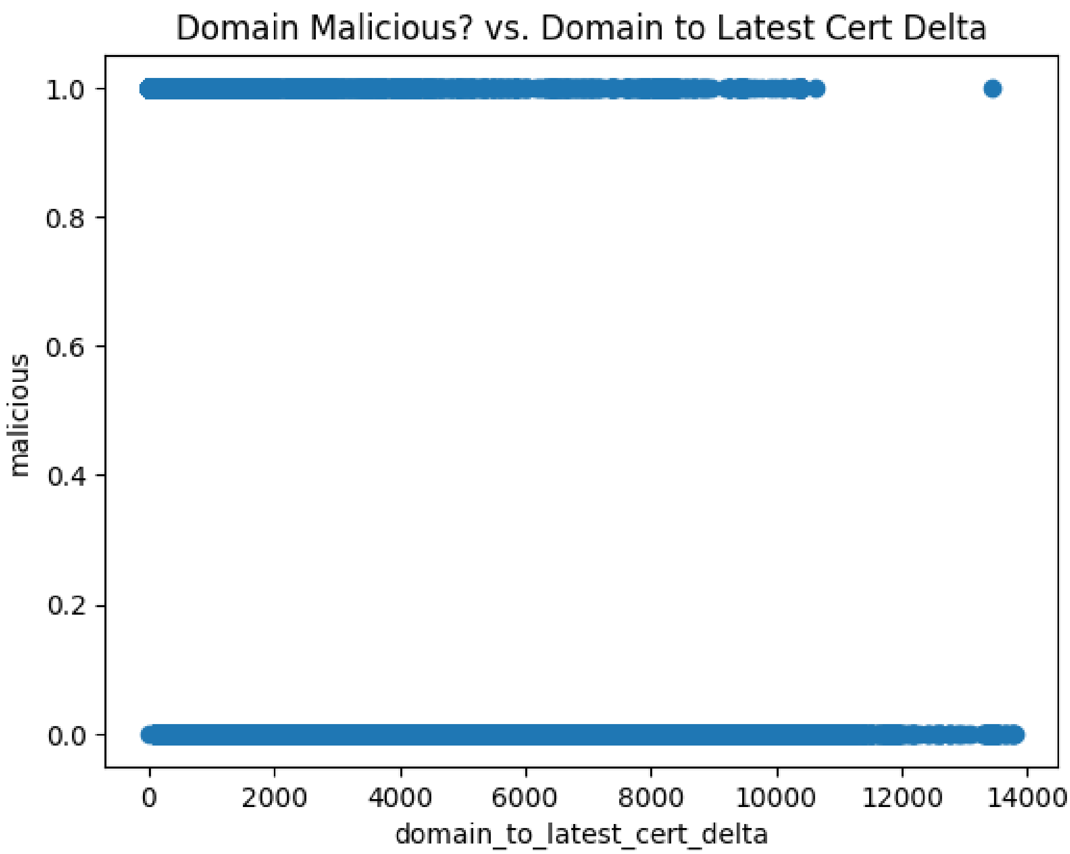


In [4]:

```
# Plot histograms
df.hist(figsize=(12,12), xrot=-45)

# Create a scatter plot of the data, showing malicious and benign domains
# plot the data as a scatter plot
plt.scatter(df["domain_to_earliest_cert_delta"], df["malicious"])
plt.xlabel("domain_to_earliest_cert_delta")
plt.ylabel("malicious")
plt.title("Domain Malicious? vs. Domain to Earliest Cert Delta")
plt.show()
# and for the latest
plt.scatter(df["domain_to_latest_cert_delta"], df["malicious"])
plt.xlabel("domain_to_latest_cert_delta")
plt.ylabel("malicious")
plt.title("Domain Malicious? vs. Domain to Latest Cert Delta")
plt.show()



click.echo(df.head())

# print a count of malicious and benign values
click.echo(df["malicious"].value_counts())
# deduplicate any rows, there shouldn't be any, but just in case
df = df.drop_duplicates()
click.echo(df["malicious"].value_counts())
```

```
click.echo(df.head())

X = df.drop(["malicious","domain", "ctlog_earliest", "ctlog_latest",
"whois_created", "whois_created_dayofweek", "ctlog_earliest_dayofweek"],
axis=1)
X = df.filter(combo_features)
y = df.filter(["malicious"])


click.echo(X.describe())
```


whois_created
ctlog_earliest
ctlog_latest
ctlog_earliest_dayofweek
ctlog_latest_dayofweek
whois_created_dow_sin
whois_created_dow_cos
ctlog_latest_dow_sin
ctlog_latest_dow_cos




```
                              domain  malicious       whois_created
0  i-db5p-cor001.api.p001.1drv.com      False 2013-08-05 18:33:50   \
4       soundcloud-pax.pandora.com      False 1993-12-28 05:00:00
5         joolcomercializadora.com       True 2023-05-22 14:53:50
6        createpdf-asr.acrobat.com      False 1999-03-16 05:00:00
8                          popt.in      False 2016-05-14 16:58:55

        ctlog_earliest        ctlog_latest  ctlog_wildcard
0 2022-01-24 20:01:58 2023-06-08 20:46:06            True   \
4 2022-05-19 00:00:00 2023-06-19 23:59:59            True
5 2022-04-06 22:23:24 2023-09-22 23:59:59           False
6 2022-09-09 00:00:00 2023-10-10 23:59:59            True
8 2023-01-07 20:36:15 2023-08-15 04:16:52           False

   whois_created_dayofweek  ctlog_earliest_dayofweek
ctlog_latest_dayofweek
0                        0                          0
3  \
4                        1                          3
0
5                        0                          2
4
6                        1                          4
1
8                        5                          5
1
```

```
   domain_to_earliest_cert_delta  domain_to_latest_cert_delta
0                         3095.0                       3595.0  \
4                        10369.0                      10766.0
5                          410.0                        124.0
6                         8578.0                       8975.0
8                         2430.0                       2649.0

   whois_created_dow_sin  whois_created_dow_cos  ctlog_earliest_dow_sin
0               0.000000               1.000000                0.000000  \
4               0.918032               0.396506               -0.340712
5               0.000000               1.000000                0.728010
6               0.918032               0.396506               -0.998199
8              -0.450871               0.892589               -0.450871

   ctlog_earliest_dow_cos  ctlog_latest_dow_sin  ctlog_latest_dow_cos
0                1.000000             -0.340712             -0.940168
4               -0.940168              0.000000              1.000000
5               -0.685567             -0.998199             -0.059997
6               -0.059997              0.918032              0.396506
8                0.892589              0.918032              0.396506
malicious
False    11739
True      9810
Name: count, dtype: int64
malicious
False    11739
True      9810
Name: count, dtype: int64
                            domain  malicious       whois_created
0  i-db5p-cor001.api.p001.1drv.com      False 2013-08-05 18:33:50  \
4       soundcloud-pax.pandora.com      False 1993-12-28 05:00:00
5         joolcomercializadora.com       True 2023-05-22 14:53:50
6        createpdf-asr.acrobat.com      False 1999-03-16 05:00:00
8                          popt.in      False 2016-05-14 16:58:55

       ctlog_earliest        ctlog_latest  ctlog_wildcard
0 2022-01-24 20:01:58 2023-06-08 20:46:06            True  \
4 2022-05-19 00:00:00 2023-06-19 23:59:59            True
5 2022-04-06 22:23:24 2023-09-22 23:59:59           False
6 2022-09-09 00:00:00 2023-10-10 23:59:59            True
8 2023-01-07 20:36:15 2023-08-15 04:16:52           False

   whois_created_dayofweek  ctlog_earliest_dayofweek
ctlog_latest_dayofweek
0                        0                         0
3  \
4                        1                         3
0
```

```
5                        0                          2
4
6                        1                          4
1
8                        5                          5
1

   domain_to_earliest_cert_delta  domain_to_latest_cert_delta
0                         3095.0                       3595.0  \
4                        10369.0                      10766.0
5                          410.0                        124.0
6                         8578.0                       8975.0
8                         2430.0                       2649.0

   whois_created_dow_sin  whois_created_dow_cos  ctlog_earliest_dow_sin
0               0.000000               1.000000                0.000000  \
4               0.918032               0.396506               -0.340712
5               0.000000               1.000000                0.728010
6               0.918032               0.396506               -0.998199
8              -0.450871               0.892589               -0.450871

   ctlog_earliest_dow_cos  ctlog_latest_dow_sin  ctlog_latest_dow_cos
0                1.000000             -0.340712             -0.940168
4               -0.940168              0.000000              1.000000
5               -0.685567             -0.998199             -0.059997
6               -0.059997              0.918032              0.396506
8                0.892589              0.918032              0.396506
       domain_to_earliest_cert_delta  domain_to_latest_cert_delta
count                   21549.000000                 21549.000000
mean                     3742.948397                  3969.491206
std                      3694.584062                  3850.835626
min                         0.000000                     0.000000
25%                       181.000000                   144.000000
50%                      2637.000000                  3009.000000
75%                      7078.000000                  7421.000000
max                     13445.000000                 13798.000000
```

In [5]:

```
# convert y (malicious) to 1/0 int
y = y.astype('int')
if "ctlog_wildcard" in X.columns:
    X["ctlog_wildcard"] = X["ctlog_wildcard"].astype('int')

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)


# random forest model

param_grid = {
```

```
    'n_estimators': [50,100,150,200],
    'max_features': ['sqrt', 'log2'],
    'max_depth' : [2,3,4,5],
    'criterion' :['gini', 'entropy']
}
```

In [6]:

```
rf = RandomForestClassifier(random_state=42)
rf_cv = GridSearchCV(estimator=rf, param_grid=param_grid, cv=5, n_jobs=-1)
rf_cv.fit(X_train, y_train.values.ravel())
```

Out[6]:

GridSearchCV

```
GridSearchCV(cv=5, estimator=RandomForestClassifier(random_state=42),
n_jobs=-1,
             param_grid={'criterion': ['gini', 'entropy'],
                         'max_depth': [2, 3, 4, 5],
                         'max_features': ['sqrt', 'log2'],
                         'n_estimators': [50, 100, 150, 200]})
```

estimator: RandomForestClassifier

```
RandomForestClassifier(random_state=42)
```

```
RandomForestClassifier
RandomForestClassifier(random_state=42)
```

In [7]:

```
bp = rf_cv.best_params_
click.echo("Best parameters set found:")
click.echo(bp)
Best parameters set found:
{'criterion': 'gini', 'max_depth': 5, 'max_features': 'sqrt',
'n_estimators': 50}
```

In [8]:

```
rf = RandomForestClassifier(random_state=42,
max_features=bp["max_features"], n_estimators=bp["n_estimators"],
max_depth=bp["max_depth"], criterion=bp["criterion"])
```

In [9]:

```
rf.fit(X_train, y_train.values.ravel())
```

Out[9]:

```
RandomForestClassifier
RandomForestClassifier(max_depth=5, n_estimators=50, random_state=42)
```

In [ ]:

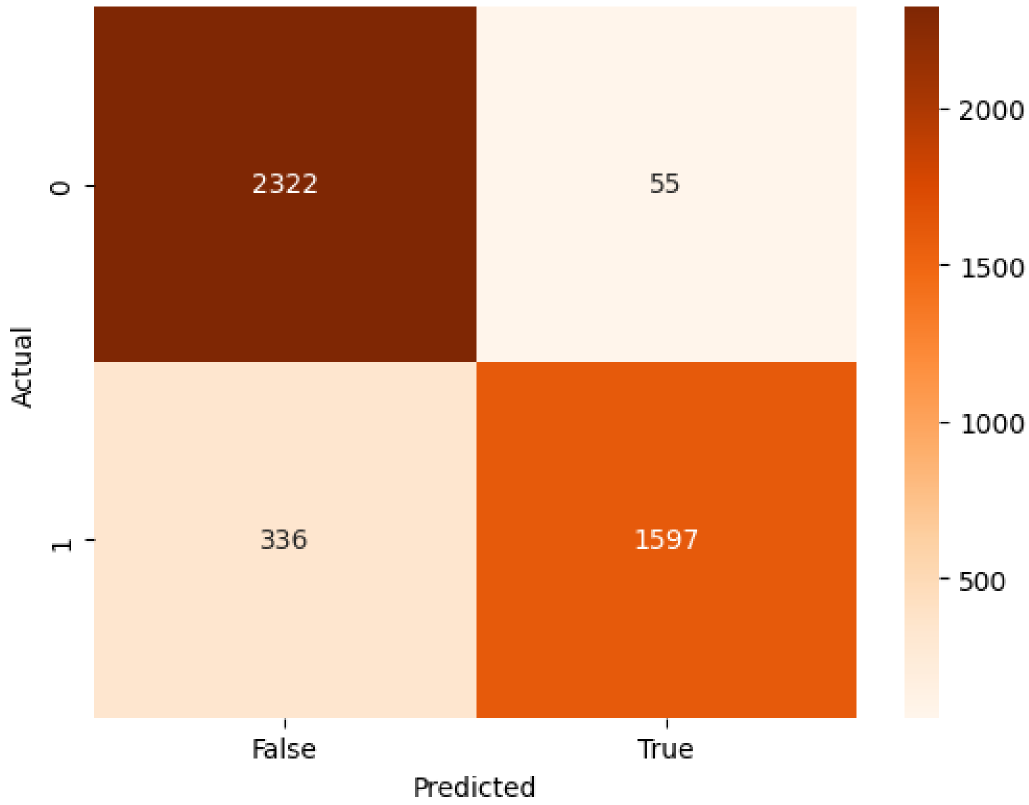
In [10]:

```
# Predict the malicious column using the test data
#add the incepts



y_predicted = rf.predict(X_test)

# Present the results
click.echo("Features selected:")
click.echo(X.columns)
```

```
click.echo("Confusion matrix:")
cm = confusion_matrix(y_test, y_predicted)
click.echo(cm)
click.echo("Classification report:")
click.echo(classification_report(y_test, y_predicted))


# Heatmap of confusion matrix
y_predicted

threshold = 0.5
y_predicted = [y > threshold for y in y_predicted]
data = {'Actual':    y_test.values.flatten(),
        'Predicted': y_predicted
        }

# Generate a confusion matrix and heatmap to evaluate the Type I and Type
II errors/ FP/FN etc.
df = pd.DataFrame(data, columns=['Actual','Predicted'])
cm2 = pd.crosstab(df['Actual'], df['Predicted'], rownames=['Actual'],
colnames=['Predicted'])
fig = sns.heatmap(cm2, annot=True, cmap='Oranges', fmt='g')
fig
Features selected:
Index(['domain_to_earliest_cert_delta', 'domain_to_latest_cert_delta'],
dtype='object')
Confusion matrix:
[[2322   55]
 [ 336 1597]]
Classification report:
              precision    recall  f1-score   support

           0       0.87      0.98      0.92      2377
           1       0.97      0.83      0.89      1933

    accuracy                           0.91      4310
   macro avg       0.92      0.90      0.91      4310
weighted avg       0.92      0.91      0.91      4310
```

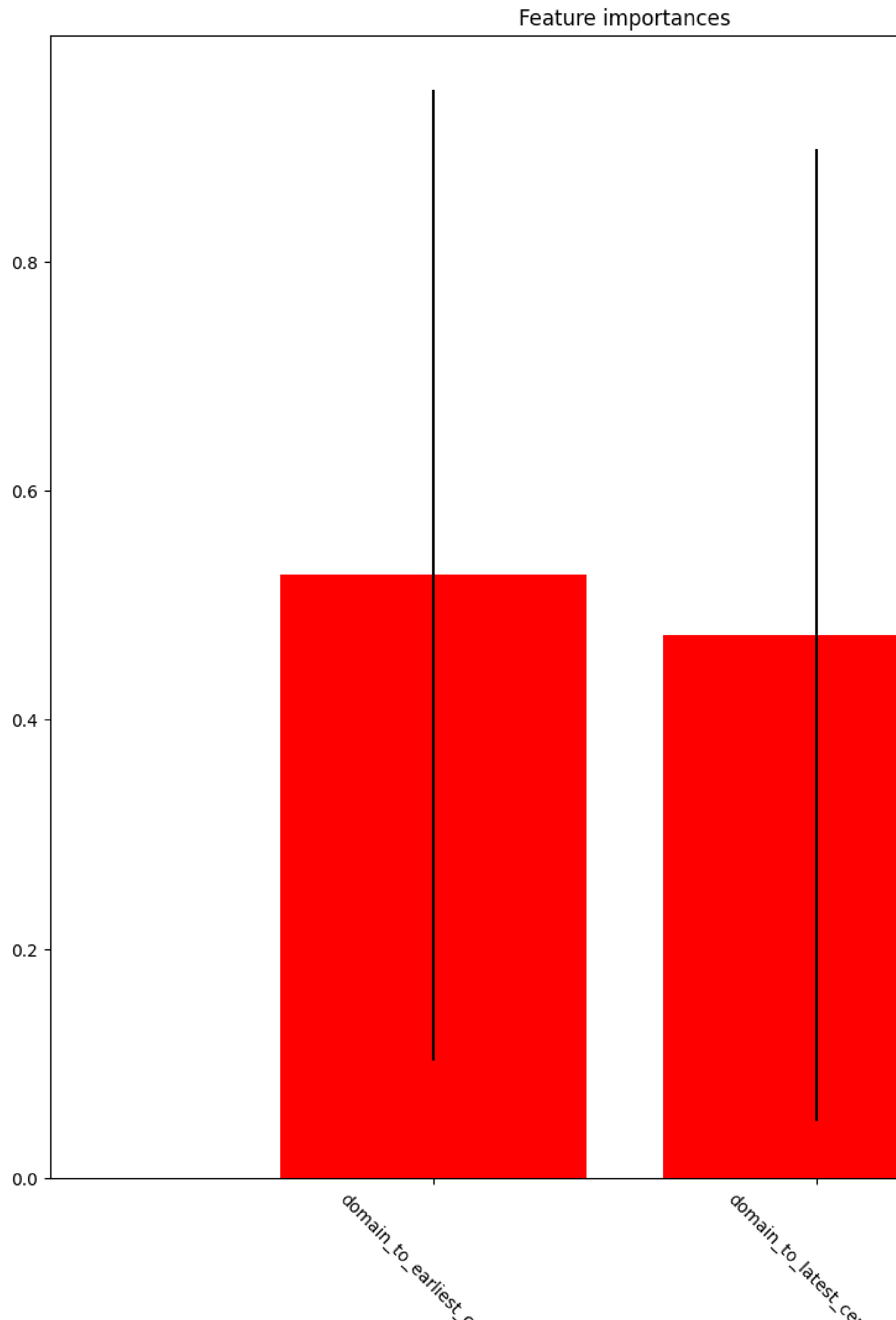
Out[10]:

```
<Axes: xlabel='Predicted', ylabel='Actual'>
```



In [11]:

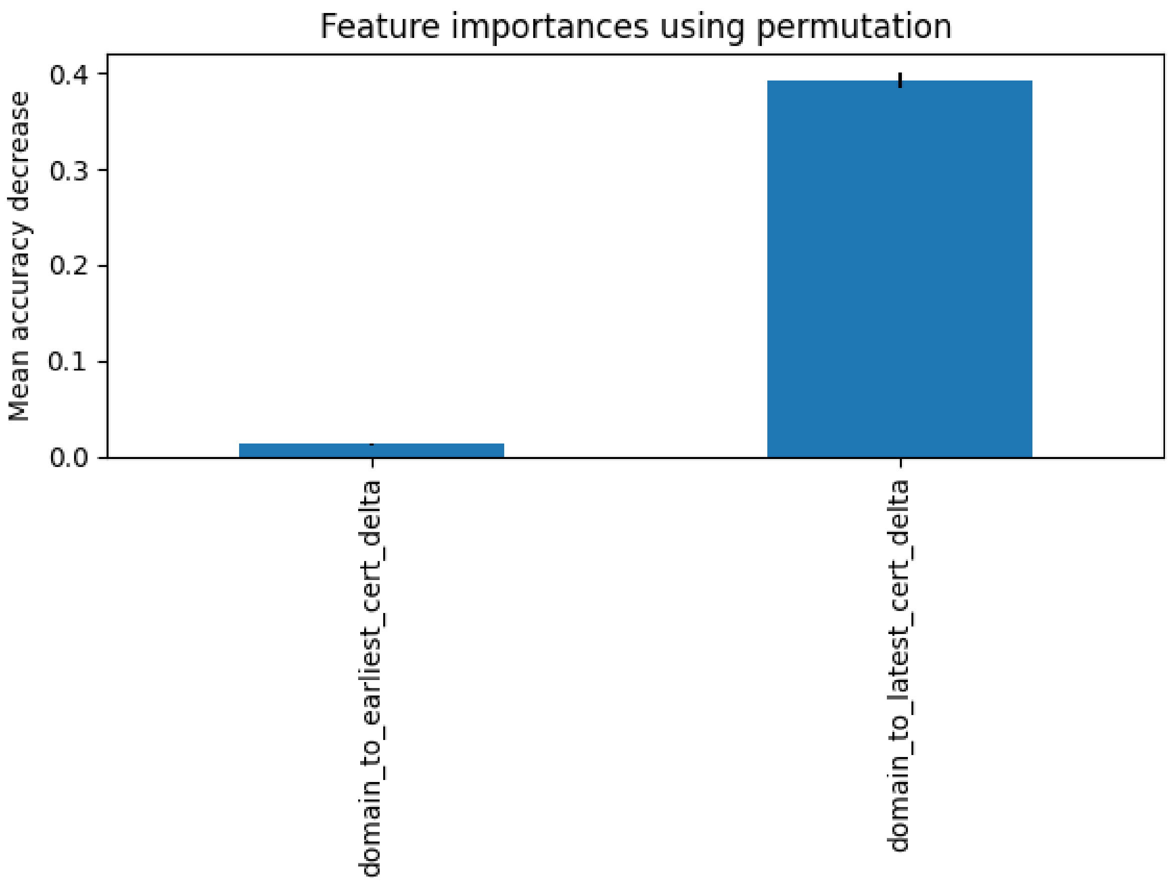
```
# plot the feature importances
importances = rf.feature_importances_
std = np.std([tree.feature_importances_ for tree in rf.estimators_],
axis=0)

indices = np.argsort(importances)[::-1]
# Print the feature ranking
click.echo("Feature ranking:")
for f in range(X.shape[1]):
    click.echo("%d. feature %s (%f)" % (f + 1, combo_features[indices[f]],
importances[indices[f]]))


# Plot the feature importances of the forest
plt.figure(figsize=(12,12))
plt.title("Feature importances")
plt.bar(range(X.shape[1]), importances[indices], color="r",
yerr=std[indices], align="center")
plt.xticks(range(X.shape[1]), X.columns[indices], rotation=-45)
plt.xlim([-1, X.shape[1]])
plt.show()
Feature ranking:
1. feature domain_to_earliest_cert_delta (0.526303)
2. feature domain_to_latest_cert_delta (0.473697)
```



Feature importances
0.8
0.6
0.4
0.2
0.0

In [12]:

```
from sklearn.inspection import permutation_importance

result = permutation_importance(rf, X_test, y_test, n_repeats=100,
random_state=42, n_jobs=-1)

forest_importances = pd.Series(result.importances_mean, index=X.columns)
fig, ax = plt.subplots()
forest_importances.plot.bar(yerr=result.importances_std, ax=ax)
ax.set_title("Feature importances using permutation")
ax.set_ylabel("Mean accuracy decrease")
fig.tight_layout()
plt.show()
```



In [ ]:

VI. Feature Set E

In [1]:

```
import click
import pandas as pd
import glob
import os
#import sklearn
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.metrics import accuracy_score, f1_score, confusion_matrix,
classification_report, roc_auc_score, roc_curve, auc
from sklearn.preprocessing import StandardScaler
from scipy import stats
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
import statsmodels.api as sm
import matplotlib.pyplot as plt
from datetime import datetime, date
# qqplot of the data
import seaborn as sns
import math
import numpy as np
import utils


combo_features = [
    'domain_to_earliest_cert_delta',
    'domain_to_latest_cert_delta',
    'ctlog_earliest_dow_sin',
    'ctlog_earliest_dow_cos',
    'ctlog_latest_dow_sin',
    'ctlog_latest_dow_cos',
    'ctlog_wildcard'
    ]
path = "../data/"
filename = None

epoch = datetime(1970,1,1)
# open the training data file, from ../data/ for the most recent merged
training data file saved by prepare-training-data-parallel.py
full_path = path + "merged_20230705-104357_training_adorned-engineered.csv"

# get latest file matching the glob
if not filename:
    filename = max(glob.glob(full_path), key=os.path.getctime)
    click.echo(f"Using {filename} as training data")
    df = pd.read_csv(filename, sep=",")

# randomize the rows
df = df.sample(frac=1, random_state=42).reset_index(drop=True)
```

```
click.echo(df.shape)
click.echo(df.columns)

""" # From prepare-training-data-parallel.py:
# Columns:
url
verification_time
domain
malicious
dateadded
whois_created
soa_timestamp
ctlog_earliest
ctlog_latest
ctlog_wildcard
whois_created_dayofweek,
ctlog_earliest_dayofweek,
domain_to_cert_delta
"""


# Data cleaning - there may be some epoch values in the whois_created
#  & ct_log_earliest columns, so we need to remove those rows

# convert the whois_created and ctlog_earliest, ctlog_latest columns to
datetime

df = df.loc[df["whois_created"] != ""]
df = df.loc[df["ctlog_earliest"] != ""]
df = df.loc[df["ctlog_latest"] != ""]

# some dates are have nanoseconds in them, so we need to remove the
nanosecond part
df["whois_created"] = df["whois_created"].str.split(".", n = 1, expand =
True)[0]
# some dates has additional timezone information, so we need to remove that
df["whois_created"] = df["whois_created"].apply(lambda x: " ".join(
str(x).split(" ")[:2]))


# convert the whois_created and ct_log_earliest columns to datetime
df = df.astype({"whois_created": 'datetime64[ns]', "ctlog_earliest":
'datetime64[ns]', "ctlog_latest": 'datetime64[ns]'})
df = df.loc[df["whois_created"].ne(pd.Timestamp('1970-01-01 00:00:00'))]
df = df.loc[df["ctlog_earliest"].ne(pd.Timestamp('1970-01-01 00:00:00'))]
df = df.loc[df["ctlog_latest"].ne(pd.Timestamp('1970-01-01 00:00:00'))]
```

```
# malicious is a bool
df['malicious'] = df['malicious'].astype('bool')
df['domain'] = df['domain'].astype('string')
Using ../data/merged_20230705-104357_training_adorned-engineered.csv as
training data
(35438, 13)
Index(['url', 'verification_time', 'domain', 'malicious', 'dateadded',
       'whois_created', 'soa_timestamp', 'ctlog_earliest', 'ctlog_latest',
       'ctlog_wildcard', 'whois_created_dayofweek',
'ctlog_earliest_dayofweek',
       'domain_to_cert_delta'],
      dtype='object')
```

In [2]:

```
#########
# Feature engineering
#########

# remove any rows where the whois_created or ctlog_earliest columns are
null
df = df.loc[df["whois_created"].notnull()]
df = df.loc[df["ctlog_earliest"].notnull()]

# if the data is missing the "whois_created_dayofweek" or
"ctlog_earliest_dayofweek" columns, then add them
# whois created day of the week 0 = Monday, 6 = Sunday
df["whois_created_dayofweek"] = df["whois_created"].apply(
        lambda x: x.weekday() if isinstance(x, date) else None
    )

# cert valid day of the week 0 = Monday, 6 = Sunday
df["ctlog_earliest_dayofweek"] = df["ctlog_earliest"].apply(
        lambda x: x.weekday() if isinstance(x, date) else None
    )

df["ctlog_latest_dayofweek"] = df["ctlog_latest"].apply(
        lambda x: x.weekday() if isinstance(x, date) else None
    )

# set data type of the day of week columns to int
df["whois_created_dayofweek"] =
df["whois_created_dayofweek"].astype("int64")
df["ctlog_earliest_dayofweek"] =
df["ctlog_earliest_dayofweek"].astype("int64")
df["ctlog_latest_dayofweek"] = df["ctlog_latest_dayofweek"].astype("int64")

# domain to cert delta
df["domain_to_earliest_cert_delta"] = df.apply(
        lambda x: utils.get_days_delta(
            x["ctlog_earliest"],
```

```
            x["whois_created"]
            if x["whois_created"] and x["ctlog_earliest"]
            else None,
        ),
        axis=1,
    )

# set the data type of the domain_to_cert_delta column to float
df["domain_to_earliest_cert_delta"] =
df["domain_to_earliest_cert_delta"].astype("float64")

# domain to latest cert delta
df["domain_to_latest_cert_delta"] = df.apply(
        lambda x: utils.get_days_delta(
            x["ctlog_latest"],
            x["whois_created"]
            if x["whois_created"] and x["ctlog_latest"]
            else None,
        ),
        axis=1,
    )

# set the data type of the domain_to_cert_delta column to float
df["domain_to_latest_cert_delta"] =
df["domain_to_latest_cert_delta"].astype("float64")

# drop columns we imported that we will never use
df = df.drop(columns=["url", "verification_time", "dateadded",
"soa_timestamp","domain_to_cert_delta"])

click.echo(df.head())
click.echo(df.describe(include='all'))
click.echo(df.dtypes)
                             domain  malicious       whois_created
0  i-db5p-cor001.api.p001.1drv.com      False 2013-08-05 18:33:50  \
4       soundcloud-pax.pandora.com      False 1993-12-28 05:00:00
5         joolcomercializadora.com       True 2023-05-22 14:53:50
6        createpdf-asr.acrobat.com      False 1999-03-16 05:00:00
8                          popt.in      False 2016-05-14 16:58:55

       ctlog_earliest        ctlog_latest  ctlog_wildcard
0 2022-01-24 20:01:58 2023-06-08 20:46:06            True  \
4 2022-05-19 00:00:00 2023-06-19 23:59:59            True
5 2022-04-06 22:23:24 2023-09-22 23:59:59           False
6 2022-09-09 00:00:00 2023-10-10 23:59:59            True
8 2023-01-07 20:36:15 2023-08-15 04:16:52           False

   whois_created_dayofweek  ctlog_earliest_dayofweek
ctlog_latest_dayofweek
```

```
0                        0                         0
3  \
4                        1                         3
0
5                        0                         2
4
6                        1                         4
1
8                        5                         5
1

   domain_to_earliest_cert_delta  domain_to_latest_cert_delta
0                        -3095.0                      -3595.0
4                       -10369.0                     -10766.0
5                          410.0                       -124.0
6                        -8578.0                      -8975.0
8                        -2430.0                      -2649.0
                  domain malicious                 whois_created
count              21549     21549                         21549  \
unique             21536         2                           NaN
top     www.mediafire.com     False                           NaN
freq                   2     11739                           NaN
mean                 NaN       NaN  2012-10-03 12:56:32.335050496
min                  NaN       NaN            1986-01-09 00:00:00
25%                  NaN       NaN            2003-05-25 13:35:05
50%                  NaN       NaN            2015-05-07 23:56:05
75%                  NaN       NaN            2023-03-20 15:03:16
max                  NaN       NaN            2023-07-03 08:21:24
std                  NaN       NaN                           NaN

                      ctlog_earliest                   ctlog_latest
count                          21549                          21549  \
unique                           NaN                            NaN
top                              NaN                            NaN
freq                             NaN                            NaN
mean    2022-09-26 15:45:50.943570432  2023-08-14 17:49:06.400900352
min              2021-11-30 05:24:28            2023-01-01 18:42:11
25%              2022-06-24 13:47:12            2023-07-02 08:11:07
50%              2022-10-18 21:00:14            2023-08-21 21:40:11
75%              2022-12-14 00:00:00            2023-09-21 19:41:38
max              2023-06-28 04:36:22            2023-12-31 23:59:59
std                              NaN                            NaN

       ctlog_wildcard  whois_created_dayofweek  ctlog_earliest_dayofweek
count           21549             21549.000000              21549.000000  \
unique              2                      NaN                       NaN
top             False                      NaN                       NaN
freq            13032                      NaN                       NaN
mean              NaN                 2.332823                  2.399462
```

```
min               NaN              0.000000                  0.000000
25%               NaN              1.000000                  1.000000
50%               NaN              2.000000                  2.000000
75%               NaN              4.000000                  4.000000
max               NaN              6.000000                  6.000000
std               NaN              1.775043                  1.897252

        ctlog_latest_dayofweek  domain_to_earliest_cert_delta
count             21549.000000                   21549.000000  \
unique                     NaN                            NaN
top                        NaN                            NaN
freq                       NaN                            NaN
mean                  2.873080                   -3645.602070
min                   0.000000                  -13445.000000
25%                   1.000000                   -7078.000000
50%                   3.000000                   -2637.000000
75%                   5.000000                      69.000000
max                   6.000000                     524.000000
std                   2.057394                    3790.677119

        domain_to_latest_cert_delta
count                  21549.000000
unique                          NaN
top                             NaN
freq                            NaN
mean                   -3967.678222
min                   -13798.000000
25%                    -7421.000000
50%                    -3009.000000
75%                     -144.000000
max                      135.000000
std                     3852.703681
domain                                  string[python]
malicious                                         bool
whois_created                           datetime64[ns]
ctlog_earliest                          datetime64[ns]
ctlog_latest                            datetime64[ns]
ctlog_wildcard                                    bool
whois_created_dayofweek                          int64
ctlog_earliest_dayofweek                         int64
ctlog_latest_dayofweek                           int64
domain_to_earliest_cert_delta                  float64
domain_to_latest_cert_delta                    float64
dtype: object
```

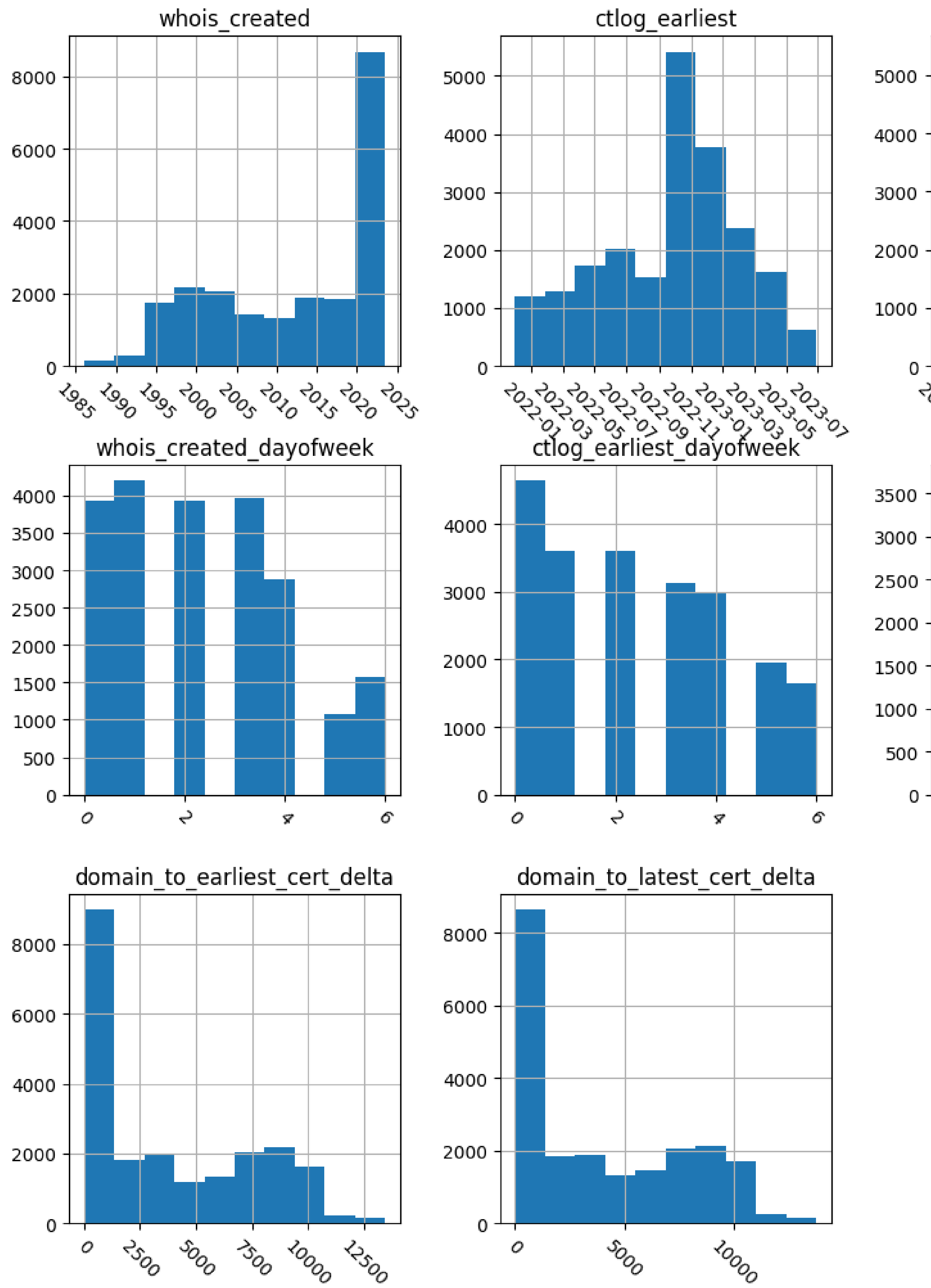
In [3]:

```
# absolute value of the domain_to_cert_delta
df["domain_to_earliest_cert_delta"] =
abs(df["domain_to_earliest_cert_delta"])
df["domain_to_latest_cert_delta"] = abs(df["domain_to_latest_cert_delta"])
```

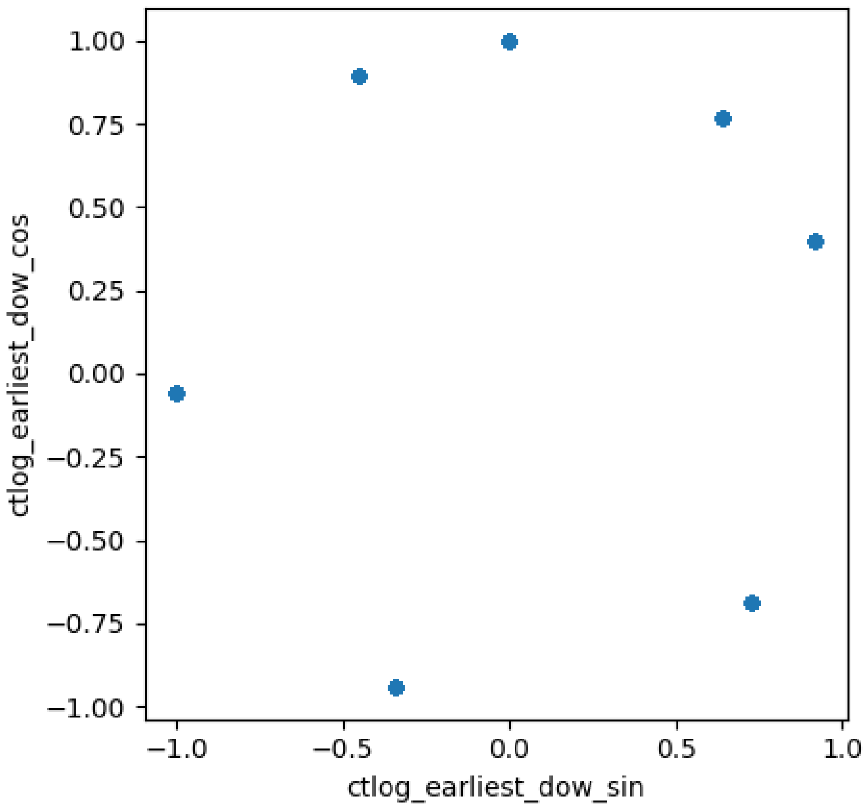
```
df.hist(figsize=(12,12), xrot=-45)

col_index = df.columns.get_loc("whois_created_dayofweek")
df["whois_created_dow_sin"] = df.apply(lambda row:
math.sin(360/7*(row[col_index])),axis=1)
df["whois_created_dow_cos"] = df.apply(lambda row:
math.cos(360/7*(row[col_index])),axis=1)

col_index = df.columns.get_loc("ctlog_earliest_dayofweek")
df["ctlog_earliest_dow_sin"] = df.apply(lambda row:
math.sin(360/7*(row[col_index])),axis=1)
df["ctlog_earliest_dow_cos"] = df.apply(lambda row:
math.cos(360/7*(row[col_index])),axis=1)

col_index = df.columns.get_loc("ctlog_latest_dayofweek")
df["ctlog_latest_dow_sin"] = df.apply(lambda row:
math.sin(360/7*(row[col_index])),axis=1)
df["ctlog_latest_dow_cos"] = df.apply(lambda row:
math.cos(360/7*(row[col_index])),axis=1)


df.plot.scatter(x="ctlog_earliest_dow_sin",
y="ctlog_earliest_dow_cos").set_aspect('equal')
df.plot.scatter(x="whois_created_dow_sin",
y="whois_created_dow_cos").set_aspect('equal')
df.plot.scatter(x="ctlog_latest_dow_sin",
y="ctlog_latest_dow_cos").set_aspect('equal')

"""
# add one hot encode ctlog_earliest_not_before_dayofweek
df = pd.get_dummies(
        df,
        columns=["ctlog_earliest_dayofweek"],
        prefix=["ctlog_earliest_dow"],
    )
# add one hot encode whois_created_dayofweek to columns
df = pd.get_dummies(
        df, columns=["whois_created_dayofweek"], prefix="whois_dow"
    )
    """

# Summary statistics
click.echo(df.describe(include='all'))
                  domain malicious                  whois_created
count              21549     21549                          21549  \
unique             21536         2                            NaN
top     www.mediafire.com     False                            NaN
freq                   2     11739                            NaN
```

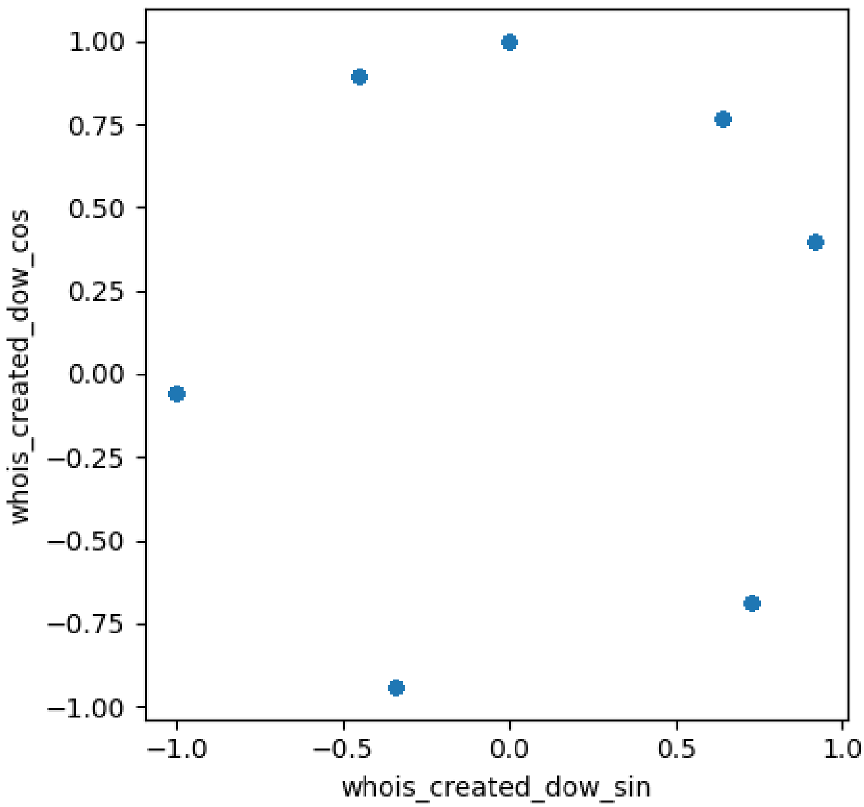
```
mean                  NaN        NaN  2012-10-03 12:56:32.335050496
min                   NaN        NaN            1986-01-09 00:00:00
25%                   NaN        NaN            2003-05-25 13:35:05
50%                   NaN        NaN            2015-05-07 23:56:05
75%                   NaN        NaN            2023-03-20 15:03:16
max                   NaN        NaN            2023-07-03 08:21:24
std                   NaN        NaN                            NaN

                      ctlog_earliest                   ctlog_latest
count                          21549                          21549  \
unique                           NaN                            NaN
top                              NaN                            NaN
freq                             NaN                            NaN
mean    2022-09-26 15:45:50.943570432  2023-08-14 17:49:06.400900352
min               2021-11-30 05:24:28            2023-01-01 18:42:11
25%               2022-06-24 13:47:12            2023-07-02 08:11:07
50%               2022-10-18 21:00:14            2023-08-21 21:40:11
75%               2022-12-14 00:00:00            2023-09-21 19:41:38
max               2023-06-28 04:36:22            2023-12-31 23:59:59
std                              NaN                            NaN

       ctlog_wildcard  whois_created_dayofweek  ctlog_earliest_dayofweek
count           21549             21549.000000              21549.000000  \
unique              2                      NaN                       NaN
top             False                      NaN                       NaN
freq            13032                      NaN                       NaN
mean              NaN                 2.332823                  2.399462
min               NaN                 0.000000                  0.000000
25%               NaN                 1.000000                  1.000000
50%               NaN                 2.000000                  2.000000
75%               NaN                 4.000000                  4.000000
max               NaN                 6.000000                  6.000000
std               NaN                 1.775043                  1.897252

        ctlog_latest_dayofweek  domain_to_earliest_cert_delta
count             21549.000000                   21549.000000  \
unique                     NaN                            NaN
top                        NaN                            NaN
freq                       NaN                            NaN
mean                  2.873080                    3742.948397
min                   0.000000                       0.000000
25%                   1.000000                     181.000000
50%                   3.000000                    2637.000000
75%                   5.000000                    7078.000000
max                   6.000000                   13445.000000
std                   2.057394                    3694.584062

        domain_to_latest_cert_delta  whois_created_dow_sin
count                  21549.000000           21549.000000  \
```

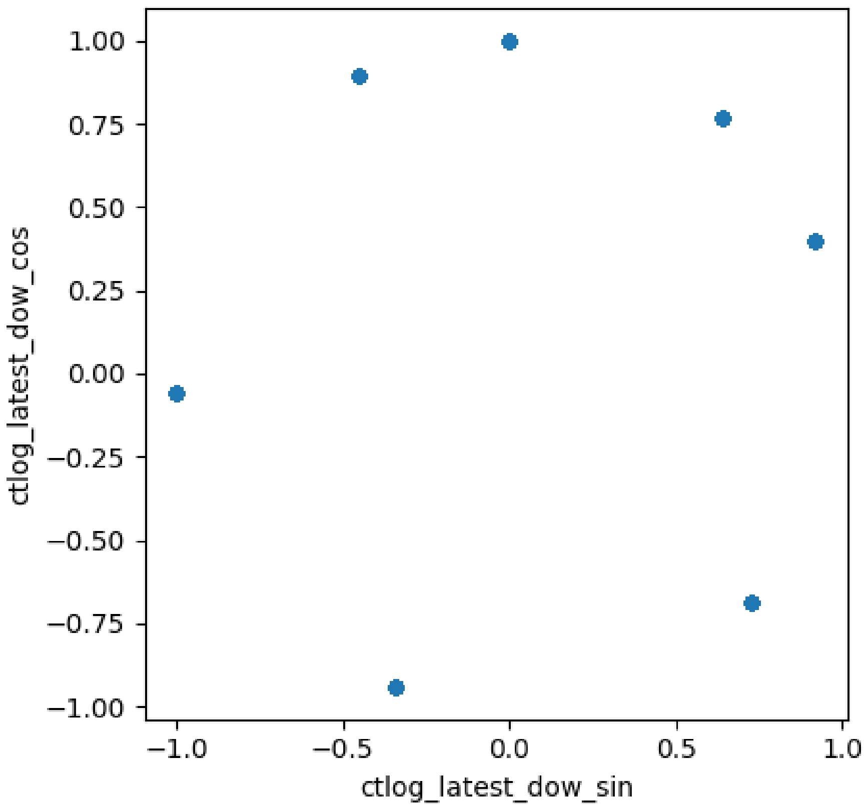
```
unique                       NaN                    NaN
top                          NaN                    NaN
freq                         NaN                    NaN
mean                 3969.491206               0.140419
min                     0.000000              -0.998199
25%                   144.000000              -0.340712
50%                  3009.000000               0.000000
75%                  7421.000000               0.728010
max                 13798.000000               0.918032
std                  3850.835626               0.659922

        whois_created_dow_cos  ctlog_earliest_dow_sin
ctlog_earliest_dow_cos
count            21549.000000            21549.000000
21549.000000  \
unique                    NaN                     NaN
NaN
top                       NaN                     NaN
NaN
freq                      NaN                     NaN
NaN
mean                 0.054288                0.095357
0.161451
min                 -0.940168               -0.998199               -
0.940168
25%                 -0.685567               -0.340712               -
0.685567
50%                  0.396506                0.000000
0.396506
75%                  0.767830                0.728010
0.892589
max                  1.000000                0.918032
1.000000
std                  0.736128                0.651782
0.734891

        ctlog_latest_dow_sin  ctlog_latest_dow_cos
count           21549.000000          21549.000000
unique                   NaN                   NaN
top                      NaN                   NaN
freq                     NaN                   NaN
mean                0.096253              0.255578
min                -0.998199             -0.940168
25%                -0.450871             -0.685567
50%                 0.000000              0.396506
75%                 0.728010              0.892589
max                 0.918032              1.000000
std                 0.651597              0.707728
```



whois_created
0
2000
4000
6000
8000
1985
1990
1995
2000
2005
2010
2015
2020
2025
ctlog_earliest
0
1000
2000
3000
4000
5000
2022-01
2022-03
2022-05
2022-07
2022-09
2022-11
2023-01
2023-03
2023-05
2023-07
whois_created_dayofweek
0
500
1000
1500
2000
2500
3000
3500
4000
0
2
4
6
ctlog_earliest_dayofweek
0
1000
2000
3000
4000
0
2
4
6
domain_to_earliest_cert_delta
0
2000
4000
6000
8000
0
2500
5000
7500
10000
12500
domain_to_latest_cert_delta
0
2000
4000
6000
8000
0
5000
10000


ctlog_earliest_dow_cos
ctlog_earliest_dow_sin
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−1.0
−0.5
0.0
0.5
1.0

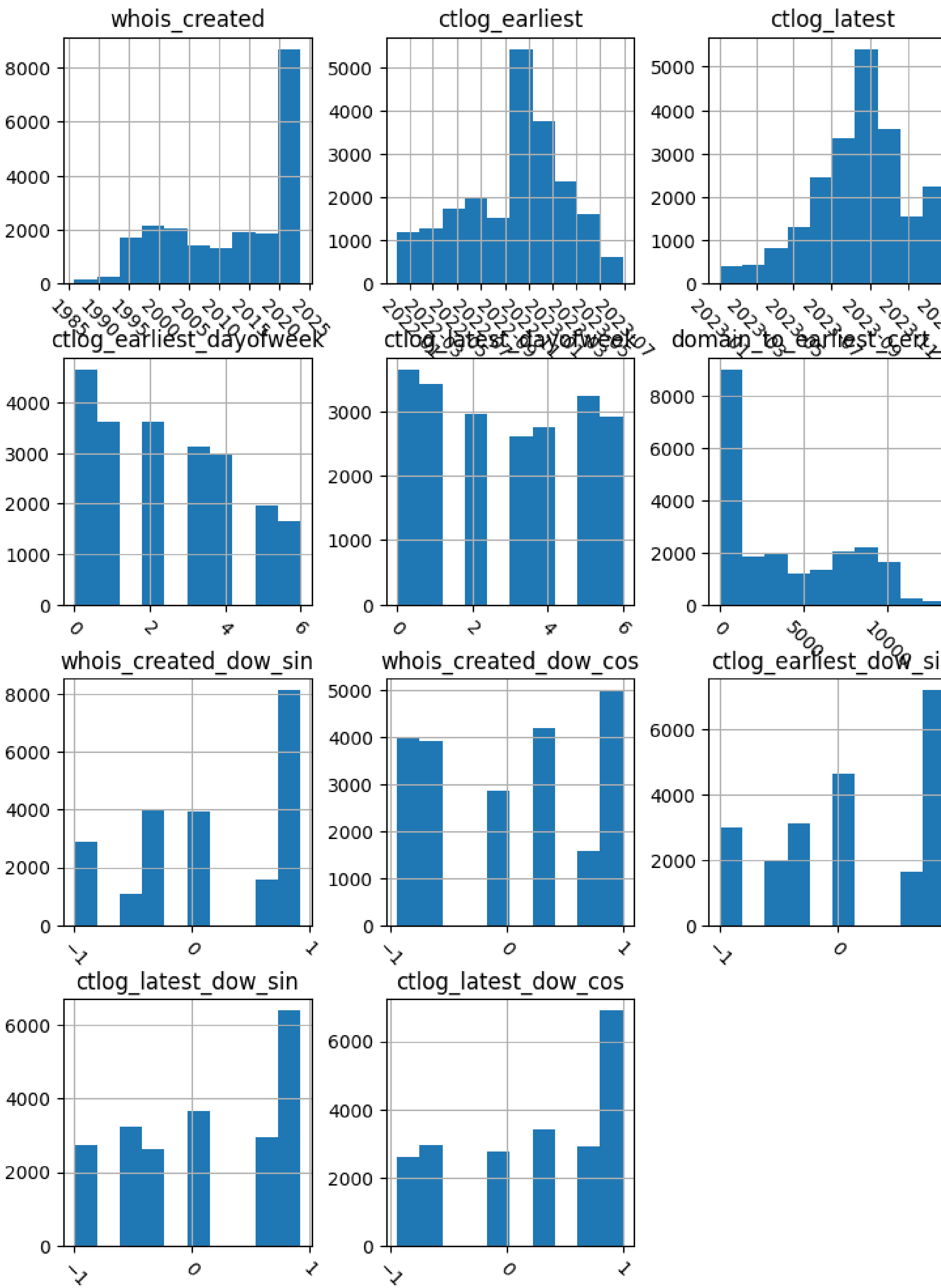

whois_created_dow_cos
whois_created_dow_sin
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−1.0
−0.5
0.0
0.5
1.0

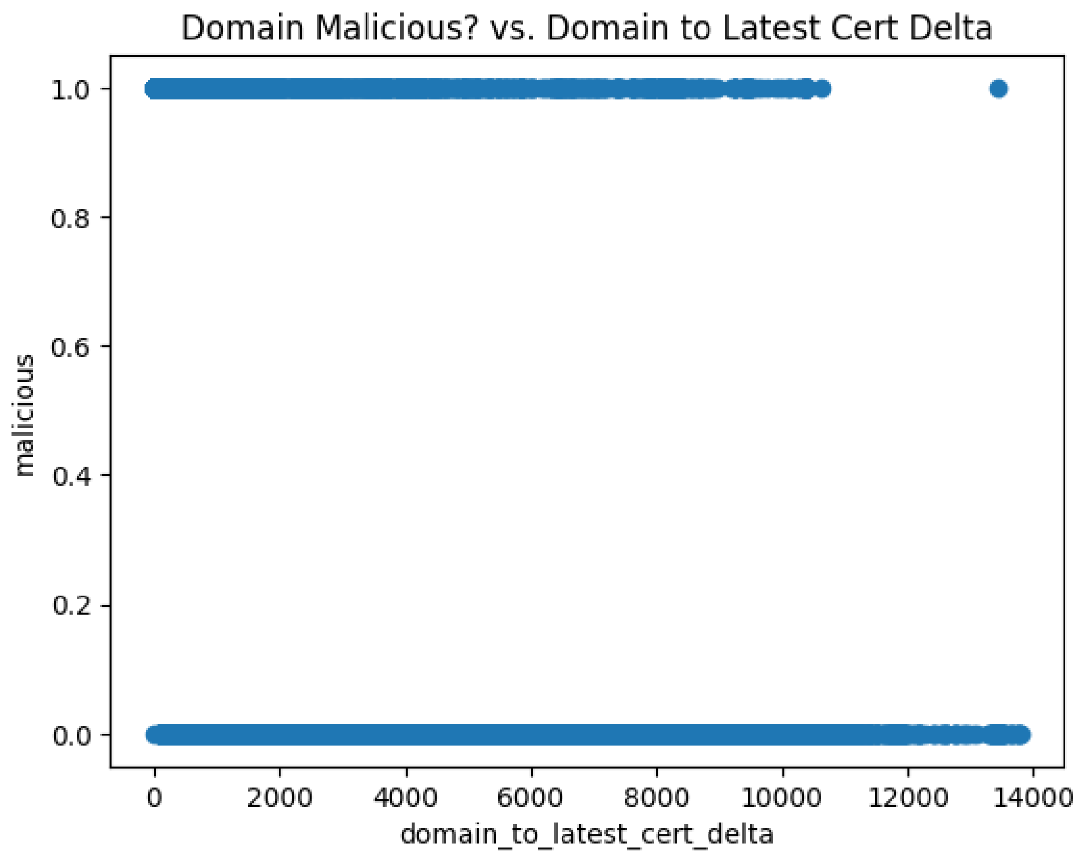


In [4]:

```
# Plot histograms
df.hist(figsize=(12,12), xrot=-45)

# Create a scatter plot of the data, showing malicious and benign domains
# plot the data as a scatter plot
plt.scatter(df["domain_to_earliest_cert_delta"], df["malicious"])
plt.xlabel("domain_to_earliest_cert_delta")
plt.ylabel("malicious")
plt.title("Domain Malicious? vs. Domain to Earliest Cert Delta")
plt.show()
# and for the latest
plt.scatter(df["domain_to_latest_cert_delta"], df["malicious"])
plt.xlabel("domain_to_latest_cert_delta")
plt.ylabel("malicious")
plt.title("Domain Malicious? vs. Domain to Latest Cert Delta")
plt.show()



click.echo(df.head())

# print a count of malicious and benign values
click.echo(df["malicious"].value_counts())
# deduplicate any rows, there shouldn't be any, but just in case
df = df.drop_duplicates()
click.echo(df["malicious"].value_counts())
```

```
click.echo(df.head())

X = df.drop(["malicious","domain", "ctlog_earliest", "ctlog_latest",
"whois_created", "whois_created_dayofweek", "ctlog_earliest_dayofweek"],
axis=1)
X = df.filter(combo_features)
y = df.filter(["malicious"])


click.echo(X.describe())
```


whois_created
ctlog_earliest
ctlog_latest
ctlog_earliest_dayofweek
ctlog_latest_dayofweek
whois_created_dow_sin
whois_created_dow_cos
ctlog_latest_dow_sin
ctlog_latest_dow_cos




```
                             domain  malicious       whois_created
0  i-db5p-cor001.api.p001.1drv.com      False 2013-08-05 18:33:50  \
4       soundcloud-pax.pandora.com      False 1993-12-28 05:00:00
5         joolcomercializadora.com       True 2023-05-22 14:53:50
6        createpdf-asr.acrobat.com      False 1999-03-16 05:00:00
8                          popt.in      False 2016-05-14 16:58:55

       ctlog_earliest        ctlog_latest  ctlog_wildcard  \
0 2022-01-24 20:01:58 2023-06-08 20:46:06            True
4 2022-05-19 00:00:00 2023-06-19 23:59:59            True
5 2022-04-06 22:23:24 2023-09-22 23:59:59           False
6 2022-09-09 00:00:00 2023-10-10 23:59:59            True
8 2023-01-07 20:36:15 2023-08-15 04:16:52           False

   whois_created_dayofweek  ctlog_earliest_dayofweek
ctlog_latest_dayofweek
0                        0                          0
3  \
4                        1                          3
0
5                        0                          2
4
6                        1                          4
1
8                        5                          5
1
```

```
   domain_to_earliest_cert_delta  domain_to_latest_cert_delta
0                         3095.0                       3595.0  \
4                        10369.0                      10766.0
5                          410.0                        124.0
6                         8578.0                       8975.0
8                         2430.0                       2649.0

   whois_created_dow_sin  whois_created_dow_cos  ctlog_earliest_dow_sin
0               0.000000               1.000000                0.000000  \
4               0.918032               0.396506               -0.340712
5               0.000000               1.000000                0.728010
6               0.918032               0.396506               -0.998199
8              -0.450871               0.892589               -0.450871

   ctlog_earliest_dow_cos  ctlog_latest_dow_sin  ctlog_latest_dow_cos
0                1.000000             -0.340712             -0.940168
4               -0.940168              0.000000              1.000000
5               -0.685567             -0.998199             -0.059997
6               -0.059997              0.918032              0.396506
8                0.892589              0.918032              0.396506
malicious
False    11739
True      9810
Name: count, dtype: int64
malicious
False    11739
True      9810
Name: count, dtype: int64
                            domain  malicious       whois_created
0  i-db5p-cor001.api.p001.1drv.com      False 2013-08-05 18:33:50  \
4       soundcloud-pax.pandora.com      False 1993-12-28 05:00:00
5         joolcomercializadora.com       True 2023-05-22 14:53:50
6        createpdf-asr.acrobat.com      False 1999-03-16 05:00:00
8                          popt.in      False 2016-05-14 16:58:55

       ctlog_earliest        ctlog_latest  ctlog_wildcard
0 2022-01-24 20:01:58 2023-06-08 20:46:06            True  \
4 2022-05-19 00:00:00 2023-06-19 23:59:59            True
5 2022-04-06 22:23:24 2023-09-22 23:59:59           False
6 2022-09-09 00:00:00 2023-10-10 23:59:59            True
8 2023-01-07 20:36:15 2023-08-15 04:16:52           False

   whois_created_dayofweek  ctlog_earliest_dayofweek
ctlog_latest_dayofweek
0                        0                         0
3  \
4                        1                         3
0
```

```
5                        0                          2
4
6                        1                          4
1
8                        5                          5
1

   domain_to_earliest_cert_delta  domain_to_latest_cert_delta
0                         3095.0                       3595.0  \
4                        10369.0                      10766.0
5                          410.0                        124.0
6                         8578.0                       8975.0
8                         2430.0                       2649.0

   whois_created_dow_sin  whois_created_dow_cos  ctlog_earliest_dow_sin
0               0.000000               1.000000                0.000000  \
4               0.918032               0.396506               -0.340712
5               0.000000               1.000000                0.728010
6               0.918032               0.396506               -0.998199
8              -0.450871               0.892589               -0.450871

   ctlog_earliest_dow_cos  ctlog_latest_dow_sin  ctlog_latest_dow_cos
0                1.000000             -0.340712             -0.940168
4               -0.940168              0.000000              1.000000
5               -0.685567             -0.998199             -0.059997
6               -0.059997              0.918032              0.396506
8                0.892589              0.918032              0.396506
       domain_to_earliest_cert_delta  domain_to_latest_cert_delta
count                   21549.000000                 21549.000000  \
mean                     3742.948397                  3969.491206
std                      3694.584062                  3850.835626
min                         0.000000                     0.000000
25%                       181.000000                   144.000000
50%                      2637.000000                  3009.000000
75%                      7078.000000                  7421.000000
max                     13445.000000                 13798.000000

       ctlog_earliest_dow_sin  ctlog_earliest_dow_cos  ctlog_latest_dow_sin
count            21549.000000            21549.000000          21549.000000
\
mean                 0.095357                0.161451              0.096253
std                  0.651782                0.734891              0.651597
min                 -0.998199               -0.940168             -0.998199
25%                 -0.340712               -0.685567             -0.450871
50%                  0.000000                0.396506              0.000000
75%                  0.728010                0.892589              0.728010
max                  0.918032                1.000000              0.918032

       ctlog_latest_dow_cos
```

```
count          21549.000000
mean               0.255578
std                0.707728
min               -0.940168
25%               -0.685567
50%                0.396506
75%                0.892589
max                1.000000
```

In [5]:

```
# convert y (malicious) to 1/0 int
y = y.astype('int')
# convert X["ctlog_wildcard"]  to 1/0 int
if "ctlog_wildcard" in X.columns:
    X["ctlog_wildcard"] = X["ctlog_wildcard"].astype('int')

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)


# random forest model

param_grid = {
    'n_estimators': [50,100,150,200],
    'max_features': ['sqrt', 'log2'],
    'max_depth' : [2,3,4,5],
    'criterion' :['gini', 'entropy']
}
```

In [6]:

```
rf = RandomForestClassifier(random_state=42)
rf_cv = GridSearchCV(estimator=rf, param_grid=param_grid, cv=5, n_jobs=-1)
rf_cv.fit(X_train, y_train.values.ravel())
```

Out[6]:

GridSearchCV

```
GridSearchCV(cv=5, estimator=RandomForestClassifier(random_state=42),
n_jobs=-1,
             param_grid={'criterion': ['gini', 'entropy'],
                         'max_depth': [2, 3, 4, 5],
                         'max_features': ['sqrt', 'log2'],
                         'n_estimators': [50, 100, 150, 200]})
```

estimator: RandomForestClassifier

```
RandomForestClassifier(random_state=42)
```

RandomForestClassifier

```
RandomForestClassifier(random_state=42)
```

In [7]:

```
bp = rf_cv.best_params_
click.echo("Best parameters set found:")
click.echo(bp)
Best parameters set found:
```

```
{'criterion': 'gini', 'max_depth': 5, 'max_features': 'sqrt',
'n_estimators': 50}
```

In [8]:

```
rf = RandomForestClassifier(random_state=42,
max_features=bp["max_features"], n_estimators=bp["n_estimators"],
max_depth=bp["max_depth"], criterion=bp["criterion"])
```

In [9]:

```
rf.fit(X_train, y_train.values.ravel())
```

Out[9]:

```
RandomForestClassifier
RandomForestClassifier(max_depth=5, n_estimators=50, random_state=42)
```

In [ ]:

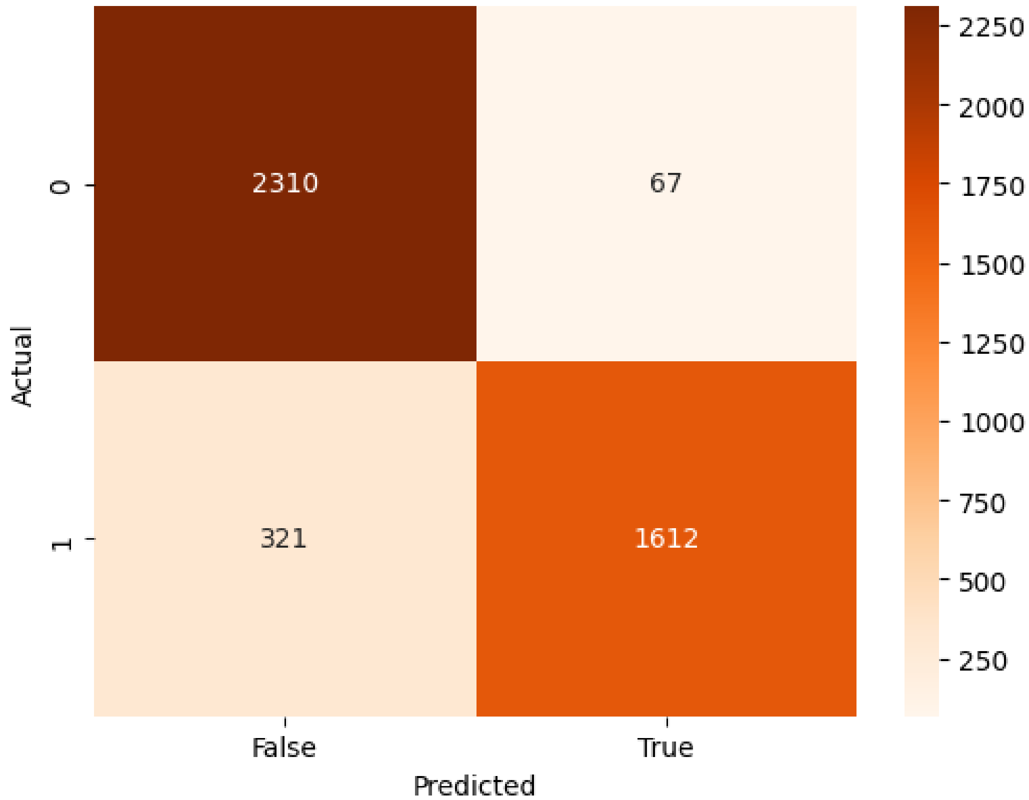
In [10]:

```
# Predict the malicious column using the test data
#add the incepts


y_predicted = rf.predict(X_test)

# Present the results
click.echo("Features selected:")
click.echo(X.columns)
click.echo("Confusion matrix:")
cm = confusion_matrix(y_test, y_predicted)
click.echo(cm)
click.echo("Classification report:")
click.echo(classification_report(y_test, y_predicted))


# Heatmap of confusion matrix
y_predicted

threshold = 0.5
y_predicted = [y > threshold for y in y_predicted]
data = {'Actual':    y_test.values.flatten(),
        'Predicted': y_predicted
        }

# Generate a confusion matrix and heatmap to evaluate the Type I and Type
II errors/ FP/FN etc.
df = pd.DataFrame(data, columns=['Actual','Predicted'])
cm2 = pd.crosstab(df['Actual'], df['Predicted'], rownames=['Actual'],
colnames=['Predicted'])
fig = sns.heatmap(cm2, annot=True, cmap='Oranges', fmt='g')
fig
Features selected:
Index(['domain_to_earliest_cert_delta', 'domain_to_latest_cert_delta',
       'ctlog_earliest_dow_sin', 'ctlog_earliest_dow_cos',
       'ctlog_latest_dow_sin', 'ctlog_latest_dow_cos', 'ctlog_wildcard'],
```

```
      dtype='object')
Confusion matrix:
[[2310   67]
 [ 321 1612]]
Classification report:
              precision    recall  f1-score   support

           0       0.88      0.97      0.92      2377
           1       0.96      0.83      0.89      1933

    accuracy                           0.91      4310
   macro avg       0.92      0.90      0.91      4310
weighted avg       0.91      0.91      0.91      4310
```

Out[10]:

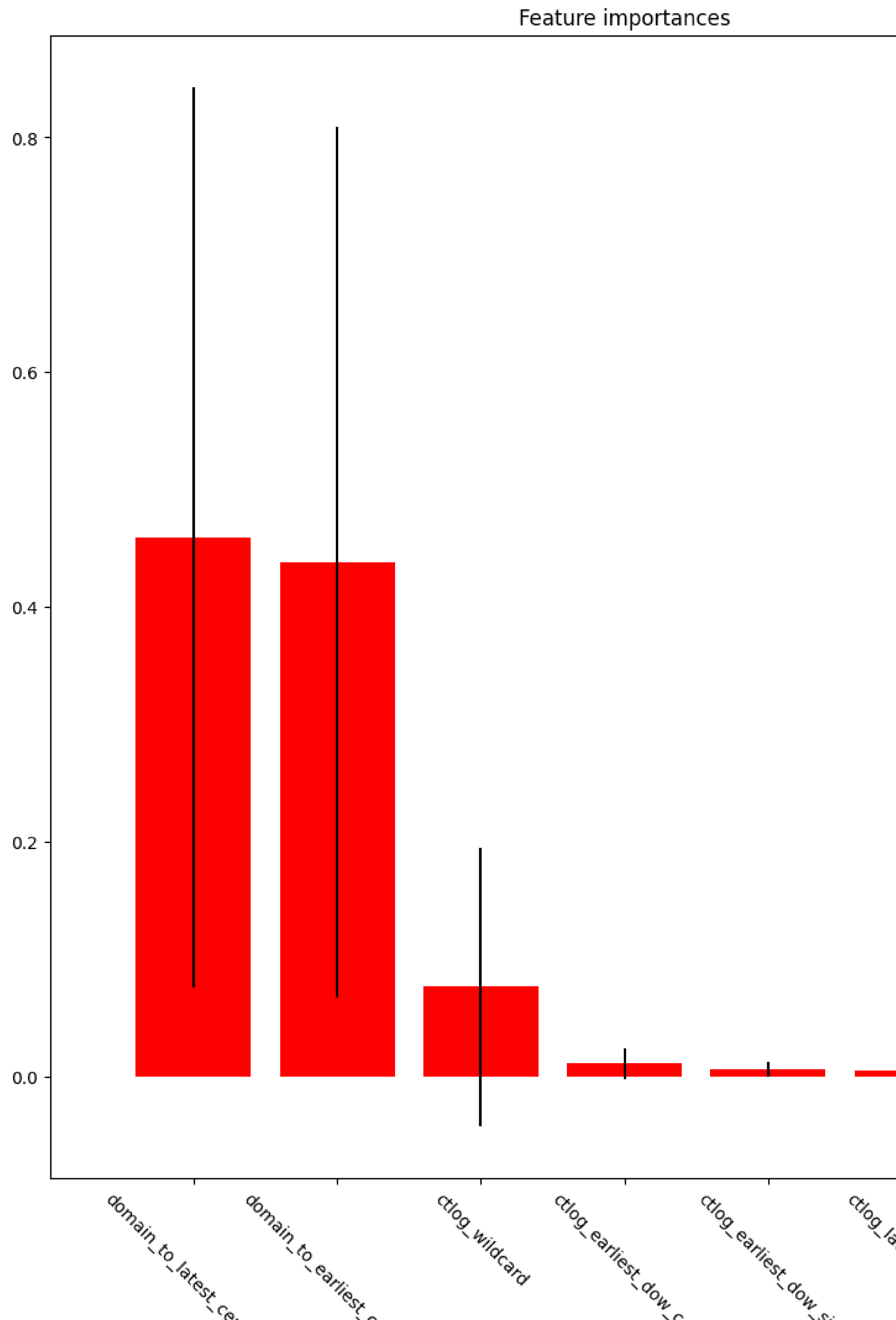
```
<Axes: xlabel='Predicted', ylabel='Actual'>
```



In [11]:

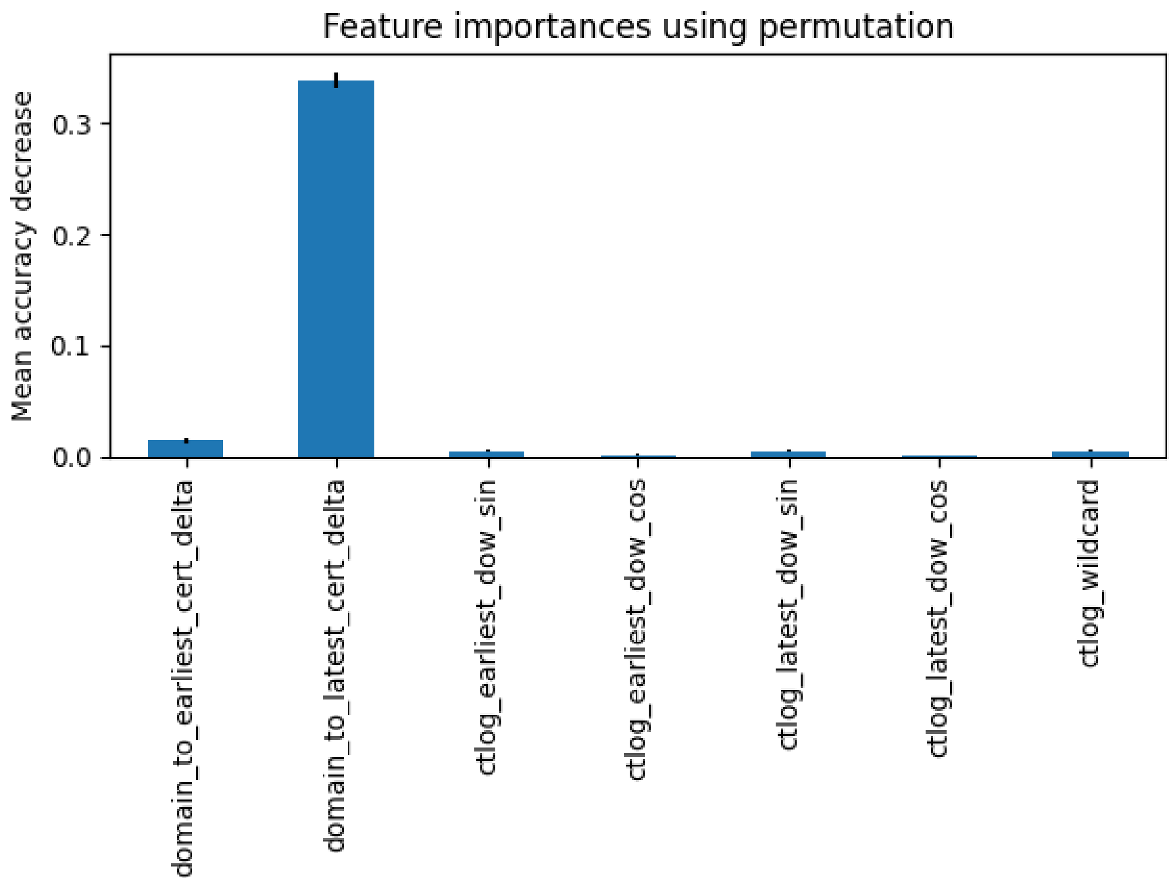
```
# plot the feature importances
importances = rf.feature_importances_
std = np.std([tree.feature_importances_ for tree in rf.estimators_],
axis=0)

indices = np.argsort(importances)[::-1]
# Print the feature ranking
click.echo("Feature ranking:")
for f in range(X.shape[1]):
```

```
    click.echo("%d. feature %s (%f)" % (f + 1, combo_features[indices[f]], 
importances[indices[f]]))

# Plot the feature importances of the forest
plt.figure(figsize=(12,12))
plt.title("Feature importances")
plt.bar(range(X.shape[1]), importances[indices], color="r", 
yerr=std[indices], align="center")
plt.xticks(range(X.shape[1]), X.columns[indices], rotation=-45)
plt.xlim([-1, X.shape[1]])
plt.show()
Feature ranking:
1. feature domain_to_latest_cert_delta (0.459260)
2. feature domain_to_earliest_cert_delta (0.437931)
3. feature ctlog_wildcard (0.076240)
4. feature ctlog_earliest_dow_cos (0.011007)
5. feature ctlog_earliest_dow_sin (0.006216)
6. feature ctlog_latest_dow_sin (0.004837)
7. feature ctlog_latest_dow_cos (0.004510)
```


Feature importances
0.8
0.6
0.4
0.2
0.0
domain_to_latest_ce
domain_to_earliest_c
ctlog_wildcard
ctlog_earliest_dow_c
ctlog_earliest_dow_s
ctlog_la

In [12]:

```
from sklearn.inspection import permutation_importance

result = permutation_importance(rf, X_test, y_test, n_repeats=100,
random_state=42, n_jobs=-1)

forest_importances = pd.Series(result.importances_mean, index=X.columns)
fig, ax = plt.subplots()
forest_importances.plot.bar(yerr=result.importances_std, ax=ax)
ax.set_title("Feature importances using permutation")
ax.set_ylabel("Mean accuracy decrease")
fig.tight_layout()
plt.show()
```



VII. Feature Set F

In [1]:

```
import click
import pandas as pd
import glob
import os
#import sklearn
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.metrics import accuracy_score, f1_score, confusion_matrix,
classification_report, roc_auc_score, roc_curve, auc
from sklearn.preprocessing import StandardScaler
from scipy import stats
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
import statsmodels.api as sm
import matplotlib.pyplot as plt
from datetime import datetime, date
# qqplot of the data
import seaborn as sns
import math
import numpy as np
import utils

combo_features =     ['domain_to_earliest_cert_delta',
'whois_created_dow_sin', 'whois_created_dow_cos']

path = "../data/"
filename = None

epoch = datetime(1970,1,1)
# open the training data file, from ../data/ for the most recent merged
training data file saved by prepare-training-data-parallel.py
full_path = path + "merged_20230705-104357_training_adorned-engineered.csv"

# get latest file matching the glob
if not filename:
    filename = max(glob.glob(full_path), key=os.path.getctime)
    click.echo(f"Using {filename} as training data")
    df = pd.read_csv(filename, sep=",")

# randomize the rows
df = df.sample(frac=1, random_state=42).reset_index(drop=True)

click.echo(df.shape)
click.echo(df.columns)

""" # From prepare-training-data-parallel.py:
# Columns:
url
```

```
verification_time
domain
malicious
dateadded
whois_created
soa_timestamp
ctlog_earliest
ctlog_latest
ctlog_wildcard
whois_created_dayofweek,
ctlog_earliest_dayofweek,
domain_to_cert_delta
"""


# Data cleaning - there may be some epoch values in the whois_created
#  & ct_log_earliest columns, so we need to remove those rows

# convert the whois_created and ctlog_earliest, ctlog_latest columns to
datetime

df = df.loc[df["whois_created"] != ""]
df = df.loc[df["ctlog_earliest"] != ""]
df = df.loc[df["ctlog_latest"] != ""]

# some dates are have nanoseconds in them, so we need to remove the
nanosecond part
df["whois_created"] = df["whois_created"].str.split(".", n = 1, expand =
True)[0]
# some dates has additional timezone information, so we need to remove that
df["whois_created"] = df["whois_created"].apply(lambda x: " ".join(
str(x).split(" ")[:2]))


# convert the whois_created and ct_log_earliest columns to datetime
df = df.astype({"whois_created": 'datetime64[ns]', "ctlog_earliest":
'datetime64[ns]', "ctlog_latest": 'datetime64[ns]'})
df = df.loc[df["whois_created"].ne(pd.Timestamp('1970-01-01 00:00:00'))]
df = df.loc[df["ctlog_earliest"].ne(pd.Timestamp('1970-01-01 00:00:00'))]
df = df.loc[df["ctlog_latest"].ne(pd.Timestamp('1970-01-01 00:00:00'))]


# malicious is a bool
df['malicious'] = df['malicious'].astype('bool')
df['domain'] = df['domain'].astype('string')
Using ../data/merged_20230705-104357_training_adorned-engineered.csv as
training data
(35438, 13)
Index(['url', 'verification_time', 'domain', 'malicious', 'dateadded',
```

```
       'whois_created', 'soa_timestamp', 'ctlog_earliest', 'ctlog_latest',
       'ctlog_wildcard', 'whois_created_dayofweek',
'ctlog_earliest_dayofweek',
       'domain_to_cert_delta'],
      dtype='object')
```

In [2]:

```
#########
# Feature engineering
#########

# remove any rows where the whois_created or ctlog_earliest columns are
null
df = df.loc[df["whois_created"].notnull()]
df = df.loc[df["ctlog_earliest"].notnull()]

# if the data is missing the "whois_created_dayofweek" or
"ctlog_earliest_dayofweek" columns, then add them
# whois created day of the week 0 = Monday, 6 = Sunday
df["whois_created_dayofweek"] = df["whois_created"].apply(
        lambda x: x.weekday() if isinstance(x, date) else None
    )

# cert valid day of the week 0 = Monday, 6 = Sunday
df["ctlog_earliest_dayofweek"] = df["ctlog_earliest"].apply(
        lambda x: x.weekday() if isinstance(x, date) else None
    )

df["ctlog_latest_dayofweek"] = df["ctlog_latest"].apply(
        lambda x: x.weekday() if isinstance(x, date) else None
    )

# set data type of the day of week columns to int
df["whois_created_dayofweek"] =
df["whois_created_dayofweek"].astype("int64")
df["ctlog_earliest_dayofweek"] =
df["ctlog_earliest_dayofweek"].astype("int64")
df["ctlog_latest_dayofweek"] = df["ctlog_latest_dayofweek"].astype("int64")

# domain to cert delta
df["domain_to_earliest_cert_delta"] = df.apply(
        lambda x: utils.get_days_delta(
            x["ctlog_earliest"],
            x["whois_created"]
            if x["whois_created"] and x["ctlog_earliest"]
            else None,
        ),
        axis=1,
    )
```

```
# set the data type of the domain_to_cert_delta column to float
df["domain_to_earliest_cert_delta"] =
df["domain_to_earliest_cert_delta"].astype("float64")

# domain to latest cert delta
df["domain_to_latest_cert_delta"] = df.apply(
        lambda x: utils.get_days_delta(
            x["ctlog_latest"],
            x["whois_created"]
            if x["whois_created"] and x["ctlog_latest"]
            else None,
        ),
        axis=1,
    )

# set the data type of the domain_to_cert_delta column to float
df["domain_to_latest_cert_delta"] =
df["domain_to_latest_cert_delta"].astype("float64")

# drop columns we imported that we will never use
df = df.drop(columns=["url", "verification_time", "dateadded",
"soa_timestamp","domain_to_cert_delta"])

click.echo(df.head())
click.echo(df.describe(include='all'))
click.echo(df.dtypes)
                            domain  malicious       whois_created
0  i-db5p-cor001.api.p001.1drv.com      False 2013-08-05 18:33:50  \
4       soundcloud-pax.pandora.com      False 1993-12-28 05:00:00
5         joolcomercializadora.com       True 2023-05-22 14:53:50
6        createpdf-asr.acrobat.com      False 1999-03-16 05:00:00
8                          popt.in      False 2016-05-14 16:58:55

       ctlog_earliest        ctlog_latest  ctlog_wildcard
0 2022-01-24 20:01:58 2023-06-08 20:46:06            True  \
4 2022-05-19 00:00:00 2023-06-19 23:59:59            True
5 2022-04-06 22:23:24 2023-09-22 23:59:59           False
6 2022-09-09 00:00:00 2023-10-10 23:59:59            True
8 2023-01-07 20:36:15 2023-08-15 04:16:52           False

   whois_created_dayofweek  ctlog_earliest_dayofweek
ctlog_latest_dayofweek
0                        0                          0
3  \
4                        1                          3
0
5                        0                          2
4
```

```
6                        1                          4
1
8                        5                          5
1

   domain_to_earliest_cert_delta  domain_to_latest_cert_delta
0                        -3095.0                      -3595.0
4                       -10369.0                     -10766.0
5                          410.0                       -124.0
6                        -8578.0                      -8975.0
8                        -2430.0                      -2649.0
                  domain malicious                  whois_created
count              21549     21549                          21549  \
unique             21536         2                            NaN
top     www.mediafire.com     False                            NaN
freq                   2     11739                            NaN
mean                 NaN       NaN  2012-10-03 12:56:32.335050496
min                  NaN       NaN            1986-01-09 00:00:00
25%                  NaN       NaN            2003-05-25 13:35:05
50%                  NaN       NaN            2015-05-07 23:56:05
75%                  NaN       NaN            2023-03-20 15:03:16
max                  NaN       NaN            2023-07-03 08:21:24
std                  NaN       NaN                            NaN

                      ctlog_earliest                   ctlog_latest
count                          21549                          21549  \
unique                           NaN                            NaN
top                              NaN                            NaN
freq                             NaN                            NaN
mean    2022-09-26 15:45:50.943570432  2023-08-14 17:49:06.400900352
min               2021-11-30 05:24:28            2023-01-01 18:42:11
25%               2022-06-24 13:47:12            2023-07-02 08:11:07
50%               2022-10-18 21:00:14            2023-08-21 21:40:11
75%               2022-12-14 00:00:00            2023-09-21 19:41:38
max               2023-06-28 04:36:22            2023-12-31 23:59:59
std                              NaN                            NaN

       ctlog_wildcard  whois_created_dayofweek  ctlog_earliest_dayofweek
count           21549             21549.000000              21549.000000  \
unique              2                      NaN                       NaN
top             False                      NaN                       NaN
freq            13032                      NaN                       NaN
mean              NaN                 2.332823                  2.399462
min               NaN                 0.000000                  0.000000
25%               NaN                 1.000000                  1.000000
50%               NaN                 2.000000                  2.000000
75%               NaN                 4.000000                  4.000000
max               NaN                 6.000000                  6.000000
std               NaN                 1.775043                  1.897252
```

```
        ctlog_latest_dayofweek  domain_to_earliest_cert_delta
count               21549.000000                   21549.000000  \
unique                       NaN                            NaN
top                          NaN                            NaN
freq                         NaN                            NaN
mean                    2.873080                   -3645.602070
min                     0.000000                  -13445.000000
25%                     1.000000                   -7078.000000
50%                     3.000000                   -2637.000000
75%                     5.000000                      69.000000
max                     6.000000                     524.000000
std                     2.057394                    3790.677119

        domain_to_latest_cert_delta
count                  21549.000000
unique                          NaN
top                             NaN
freq                            NaN
mean                   -3967.678222
min                   -13798.000000
25%                    -7421.000000
50%                    -3009.000000
75%                     -144.000000
max                      135.000000
std                     3852.703681
domain                                string[python]
malicious                                       bool
whois_created                         datetime64[ns]
ctlog_earliest                        datetime64[ns]
ctlog_latest                          datetime64[ns]
ctlog_wildcard                                  bool
whois_created_dayofweek                        int64
ctlog_earliest_dayofweek                       int64
ctlog_latest_dayofweek                         int64
domain_to_earliest_cert_delta                float64
domain_to_latest_cert_delta                  float64
dtype: object
```

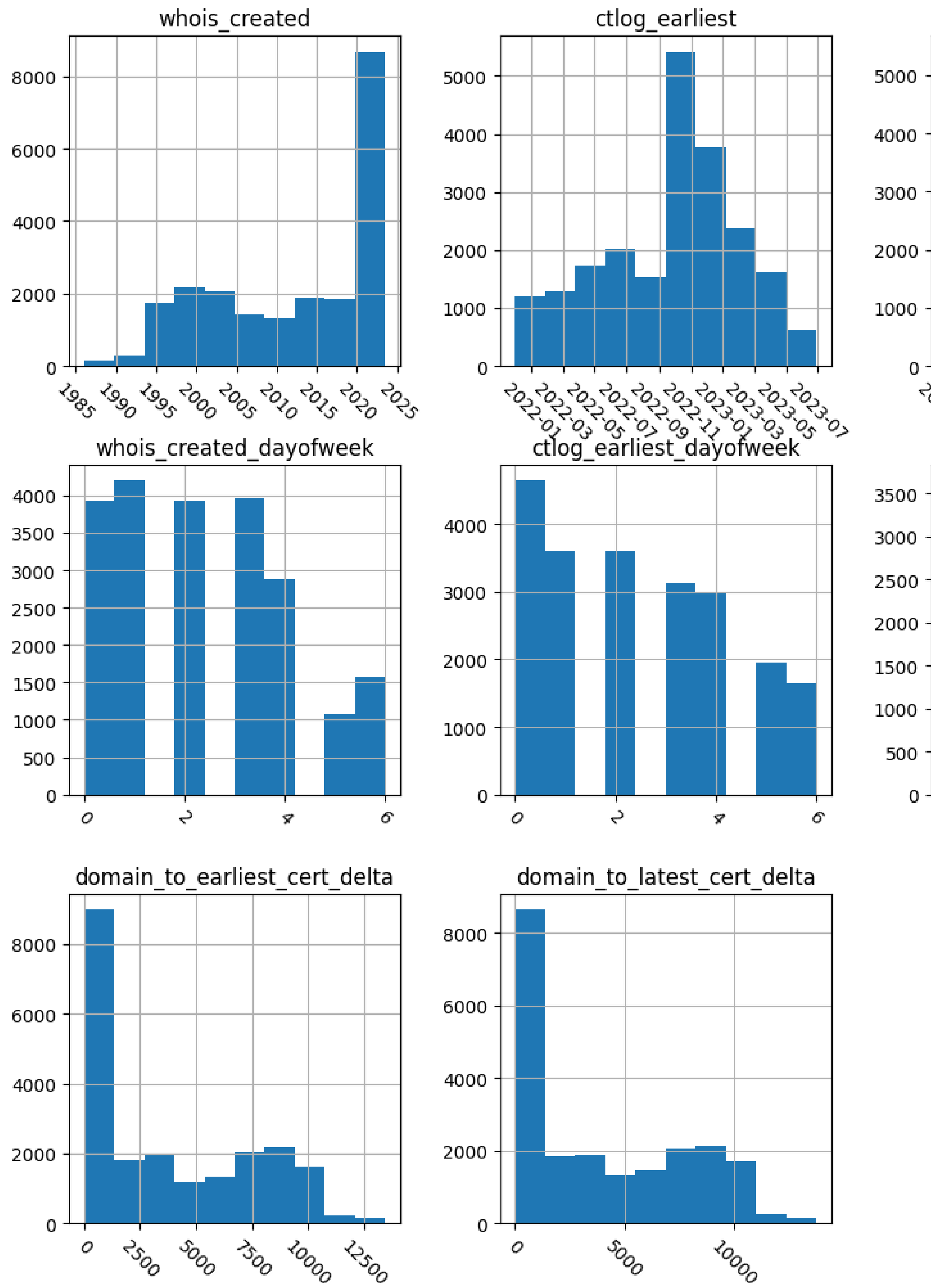
In [3]:

```
# absolute value of the domain_to_cert_delta
df["domain_to_earliest_cert_delta"] =
abs(df["domain_to_earliest_cert_delta"])
df["domain_to_latest_cert_delta"] = abs(df["domain_to_latest_cert_delta"])

df.hist(figsize=(12,12), xrot=-45)

col_index = df.columns.get_loc("whois_created_dayofweek")
df["whois_created_dow_sin"] = df.apply(lambda row:
math.sin(360/7*(row[col_index])),axis=1)
```

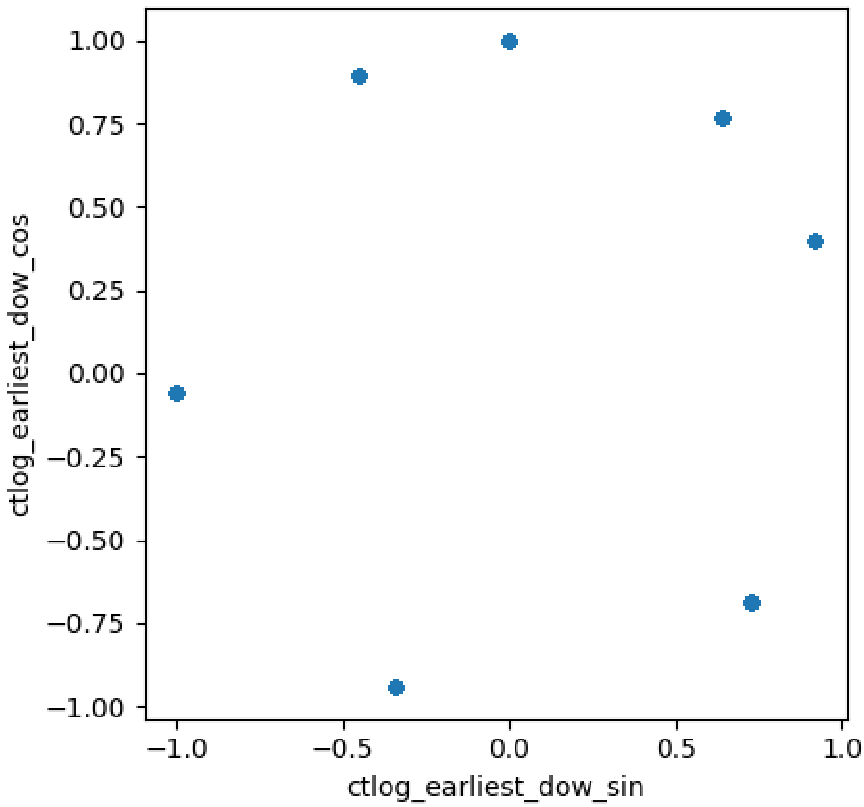
```
df["whois_created_dow_cos"] = df.apply(lambda row:
math.cos(360/7*(row[col_index])),axis=1)

col_index = df.columns.get_loc("ctlog_earliest_dayofweek")
df["ctlog_earliest_dow_sin"] = df.apply(lambda row:
math.sin(360/7*(row[col_index])),axis=1)
df["ctlog_earliest_dow_cos"] = df.apply(lambda row:
math.cos(360/7*(row[col_index])),axis=1)

col_index = df.columns.get_loc("ctlog_latest_dayofweek")
df["ctlog_latest_dow_sin"] = df.apply(lambda row:
math.sin(360/7*(row[col_index])),axis=1)
df["ctlog_latest_dow_cos"] = df.apply(lambda row:
math.cos(360/7*(row[col_index])),axis=1)


df.plot.scatter(x="ctlog_earliest_dow_sin",
y="ctlog_earliest_dow_cos").set_aspect('equal')
df.plot.scatter(x="whois_created_dow_sin",
y="whois_created_dow_cos").set_aspect('equal')
df.plot.scatter(x="ctlog_latest_dow_sin",
y="ctlog_latest_dow_cos").set_aspect('equal')

"""
# add one hot encode ctlog_earliest_not_before_dayofweek
df = pd.get_dummies(
        df,
        columns=["ctlog_earliest_dayofweek"],
        prefix=["ctlog_earliest_dow"],
    )
# add one hot encode whois_created_dayofweek to columns
df = pd.get_dummies(
        df, columns=["whois_created_dayofweek"], prefix="whois_dow"
    )
    """

# Summary statistics
click.echo(df.describe(include='all'))
                   domain malicious                    whois_created
count               21549     21549                            21549  \
unique              21536         2                              NaN
top     www.mediafire.com     False                              NaN
freq                    2     11739                              NaN
mean                  NaN       NaN  2012-10-03 12:56:32.335050496
min                   NaN       NaN            1986-01-09 00:00:00
25%                   NaN       NaN            2003-05-25 13:35:05
50%                   NaN       NaN            2015-05-07 23:56:05
75%                   NaN       NaN            2023-03-20 15:03:16
max                   NaN       NaN            2023-07-03 08:21:24
```

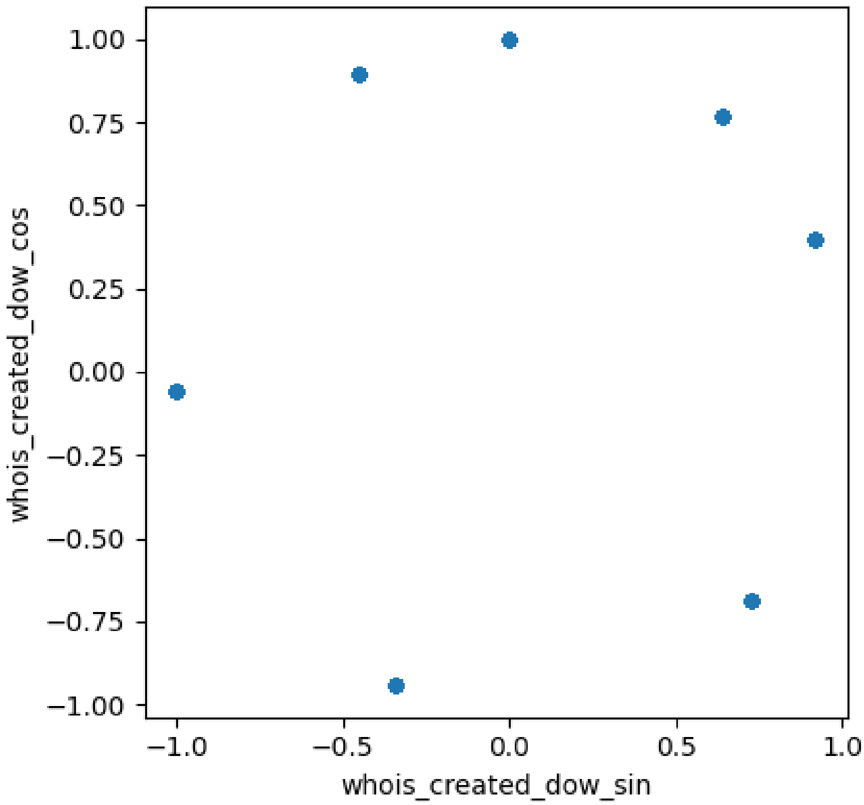
```
std                   NaN       NaN                              NaN

                  ctlog_earliest                 ctlog_latest
count                      21549                        21549  \
unique                       NaN                          NaN
top                          NaN                          NaN
freq                         NaN                          NaN
mean    2022-09-26 15:45:50.943570432  2023-08-14 17:49:06.400900352
min              2021-11-30 05:24:28          2023-01-01 18:42:11
25%              2022-06-24 13:47:12          2023-07-02 08:11:07
50%              2022-10-18 21:00:14          2023-08-21 21:40:11
75%              2022-12-14 00:00:00          2023-09-21 19:41:38
max              2023-06-28 04:36:22          2023-12-31 23:59:59
std                          NaN                          NaN

       ctlog_wildcard  whois_created_dayofweek  ctlog_earliest_dayofweek
count           21549             21549.000000              21549.000000  \
unique              2                      NaN                       NaN
top             False                      NaN                       NaN
freq            13032                      NaN                       NaN
mean              NaN                 2.332823                  2.399462
min               NaN                 0.000000                  0.000000
25%               NaN                 1.000000                  1.000000
50%               NaN                 2.000000                  2.000000
75%               NaN                 4.000000                  4.000000
max               NaN                 6.000000                  6.000000
std               NaN                 1.775043                  1.897252

        ctlog_latest_dayofweek  domain_to_earliest_cert_delta
count             21549.000000                   21549.000000  \
unique                     NaN                            NaN
top                        NaN                            NaN
freq                       NaN                            NaN
mean                  2.873080                    3742.948397
min                   0.000000                       0.000000
25%                   1.000000                     181.000000
50%                   3.000000                    2637.000000
75%                   5.000000                    7078.000000
max                   6.000000                   13445.000000
std                   2.057394                    3694.584062

        domain_to_latest_cert_delta  whois_created_dow_sin
count                  21549.000000           21549.000000  \
unique                          NaN                    NaN
top                             NaN                    NaN
freq                            NaN                    NaN
mean                    3969.491206               0.140419
min                        0.000000              -0.998199
25%                      144.000000              -0.340712
```

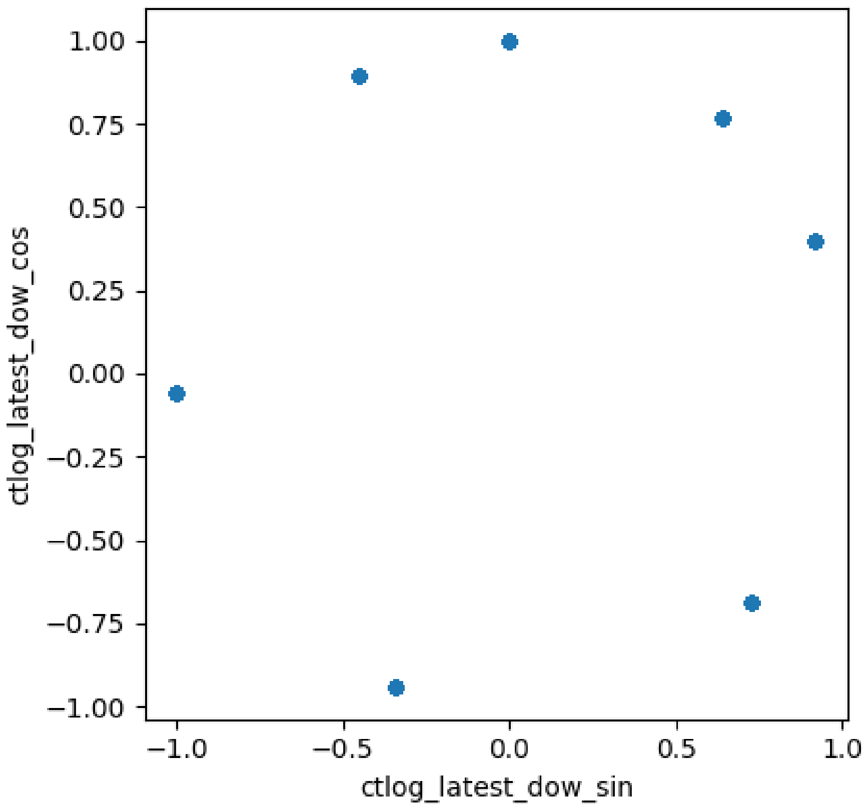
```
50%                   3009.000000              0.000000
75%                   7421.000000              0.728010
max                  13798.000000              0.918032
std                   3850.835626              0.659922

        whois_created_dow_cos  ctlog_earliest_dow_sin
ctlog_earliest_dow_cos
count            21549.000000            21549.000000
21549.000000  \
unique                    NaN                     NaN
NaN
top                       NaN                     NaN
NaN
freq                      NaN                     NaN
NaN
mean                 0.054288                0.095357
0.161451
min                 -0.940168               -0.998199               -
0.940168
25%                 -0.685567               -0.340712               -
0.685567
50%                  0.396506                0.000000
0.396506
75%                  0.767830                0.728010
0.892589
max                  1.000000                0.918032
1.000000
std                  0.736128                0.651782
0.734891

        ctlog_latest_dow_sin  ctlog_latest_dow_cos
count           21549.000000          21549.000000
unique                   NaN                   NaN
top                      NaN                   NaN
freq                     NaN                   NaN
mean                0.096253              0.255578
min                -0.998199             -0.940168
25%                -0.450871             -0.685567
50%                 0.000000              0.396506
75%                 0.728010              0.892589
max                 0.918032              1.000000
std                 0.651597              0.707728
```



whois_created
0
2000
4000
6000
8000
1985
1990
1995
2000
2005
2010
2015
2020
2025
ctlog_earliest
0
1000
2000
3000
4000
5000
2022-01
2022-03
2022-05
2022-07
2022-09
2022-11
2023-01
2023-03
2023-05
2023-07
whois_created_dayofweek
0
500
1000
1500
2000
2500
3000
3500
4000
0
2
4
6
ctlog_earliest_dayofweek
0
1000
2000
3000
4000
0
2
4
6
domain_to_earliest_cert_delta
0
2000
4000
6000
8000
0
2500
5000
7500
10000
12500
domain_to_latest_cert_delta
0
2000
4000
6000
8000
0
5000
10000


ctlog_earliest_dow_cos
ctlog_earliest_dow_sin
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−1.0
−0.5
0.0
0.5
1.0

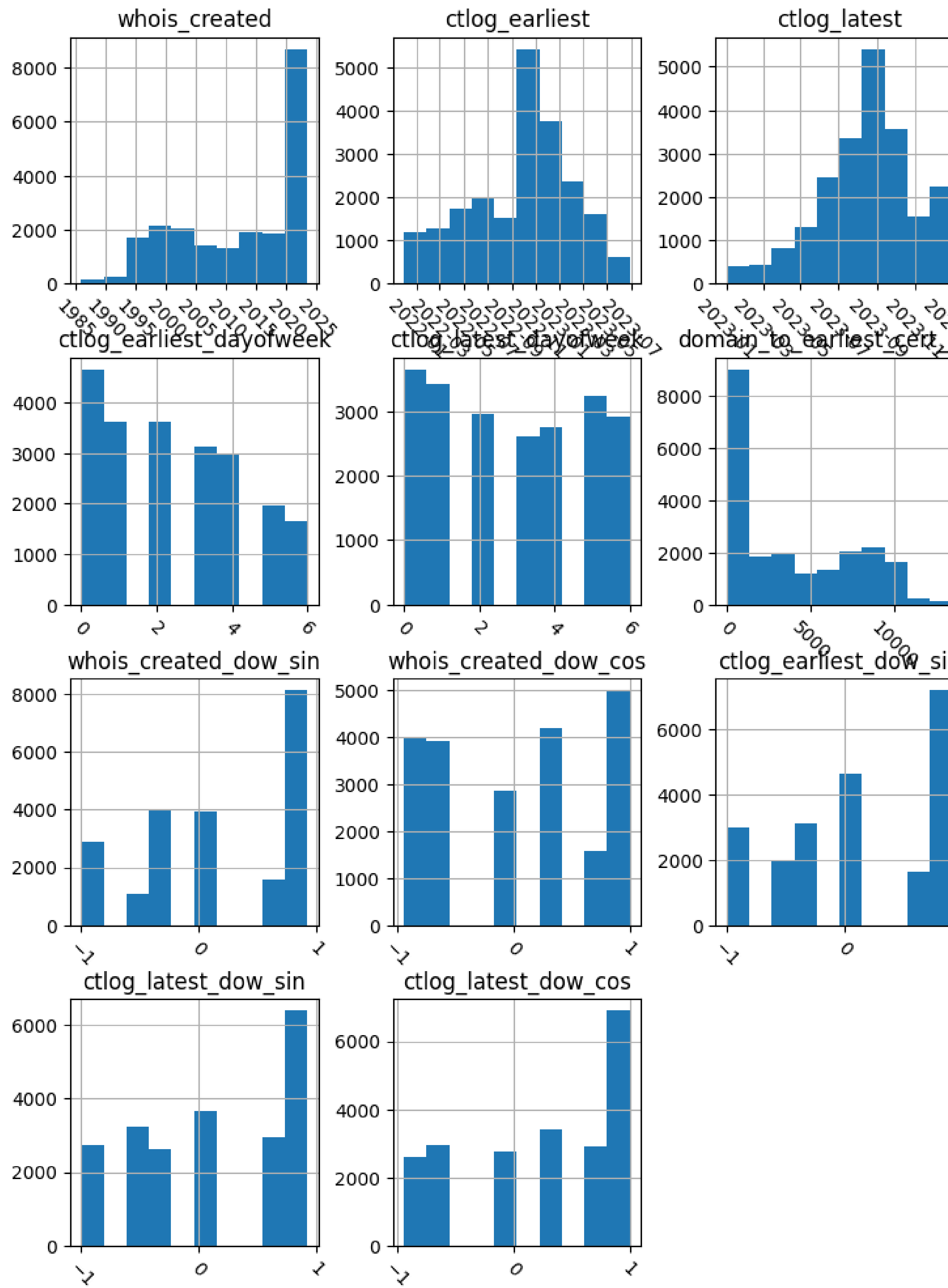

whois_created_dow_cos
whois_created_dow_sin
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−1.0
−0.5
0.0
0.5
1.0

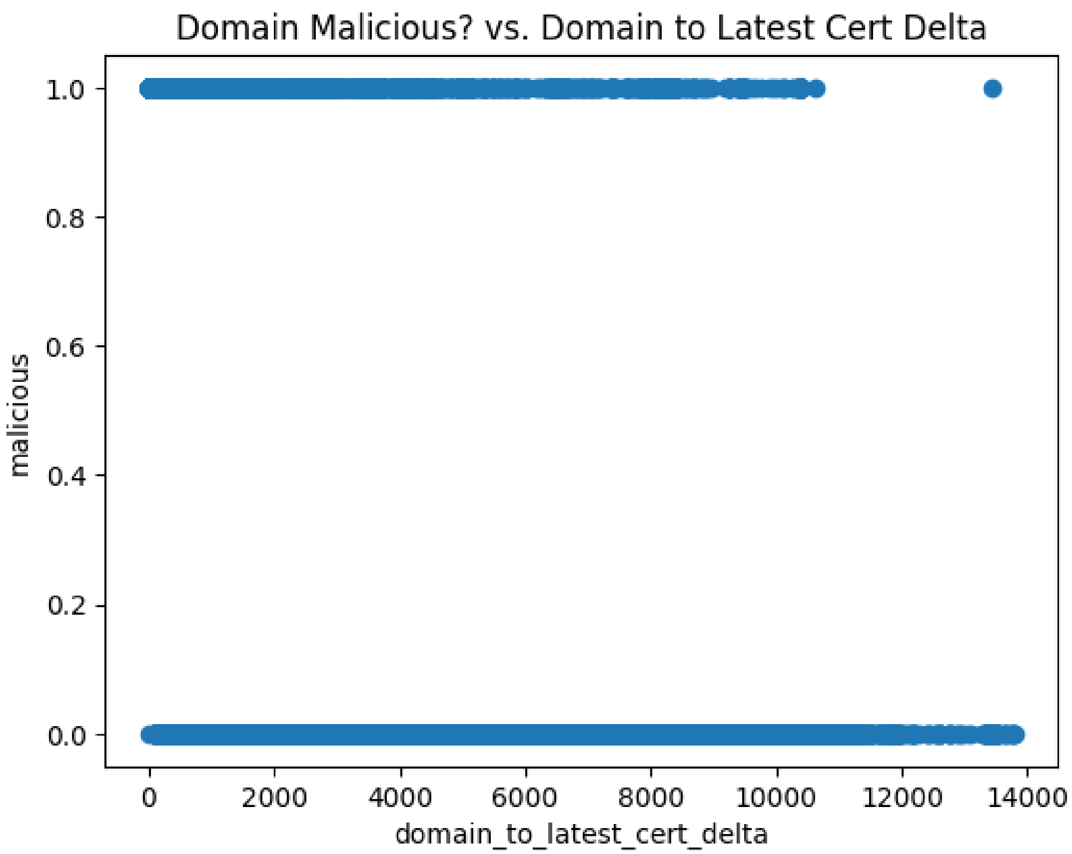


In [4]:

```
# Plot histograms
df.hist(figsize=(12,12), xrot=-45)

# Create a scatter plot of the data, showing malicious and benign domains
# plot the data as a scatter plot
plt.scatter(df["domain_to_earliest_cert_delta"], df["malicious"])
plt.xlabel("domain_to_earliest_cert_delta")
plt.ylabel("malicious")
plt.title("Domain Malicious? vs. Domain to Earliest Cert Delta")
plt.show()
# and for the latest
plt.scatter(df["domain_to_latest_cert_delta"], df["malicious"])
plt.xlabel("domain_to_latest_cert_delta")
plt.ylabel("malicious")
plt.title("Domain Malicious? vs. Domain to Latest Cert Delta")
plt.show()



click.echo(df.head())

# print a count of malicious and benign values
click.echo(df["malicious"].value_counts())
# deduplicate any rows, there shouldn't be any, but just in case
df = df.drop_duplicates()
click.echo(df["malicious"].value_counts())
```

```
click.echo(df.head())

X = df.drop(["malicious","domain", "ctlog_earliest", "ctlog_latest",
"whois_created", "whois_created_dayofweek", "ctlog_earliest_dayofweek"],
axis=1)
X = df.filter(combo_features)
y = df.filter(["malicious"])


click.echo(X.describe())
```


whois_created
ctlog_earliest
ctlog_latest
ctlog_earliest_dayofweek
ctlog_latest_dayofweek
whois_created_dow_sin
whois_created_dow_cos
ctlog_latest_dow_sin
ctlog_latest_dow_cos
0
2000
4000
6000
8000
1000
3000
5000
1985
1990
1995
2000
2005
2010
2015
2020
2025
2
4
6
5000
10000
-1
1




```
                          domain  malicious        whois_created
0  i-db5p-cor001.api.p001.1drv.com      False 2013-08-05 18:33:50   \
4       soundcloud-pax.pandora.com      False 1993-12-28 05:00:00
5         joolcomercializadora.com       True 2023-05-22 14:53:50
6        createpdf-asr.acrobat.com      False 1999-03-16 05:00:00
8                          popt.in      False 2016-05-14 16:58:55

       ctlog_earliest        ctlog_latest  ctlog_wildcard
0 2022-01-24 20:01:58 2023-06-08 20:46:06            True   \
4 2022-05-19 00:00:00 2023-06-19 23:59:59            True
5 2022-04-06 22:23:24 2023-09-22 23:59:59           False
6 2022-09-09 00:00:00 2023-10-10 23:59:59            True
8 2023-01-07 20:36:15 2023-08-15 04:16:52           False

   whois_created_dayofweek  ctlog_earliest_dayofweek
ctlog_latest_dayofweek
0                        0                          0
3  \
4                        1                          3
0
5                        0                          2
4
6                        1                          4
1
8                        5                          5
1
```

```
   domain_to_earliest_cert_delta  domain_to_latest_cert_delta
0                         3095.0                       3595.0  \
4                        10369.0                      10766.0
5                          410.0                        124.0
6                         8578.0                       8975.0
8                         2430.0                       2649.0

   whois_created_dow_sin  whois_created_dow_cos  ctlog_earliest_dow_sin
0               0.000000               1.000000                0.000000  \
4               0.918032               0.396506               -0.340712
5               0.000000               1.000000                0.728010
6               0.918032               0.396506               -0.998199
8              -0.450871               0.892589               -0.450871

   ctlog_earliest_dow_cos  ctlog_latest_dow_sin  ctlog_latest_dow_cos
0                1.000000             -0.340712             -0.940168
4               -0.940168              0.000000              1.000000
5               -0.685567             -0.998199             -0.059997
6               -0.059997              0.918032              0.396506
8                0.892589              0.918032              0.396506
malicious
False    11739
True      9810
Name: count, dtype: int64
malicious
False    11739
True      9810
Name: count, dtype: int64
                            domain  malicious       whois_created
0  i-db5p-cor001.api.p001.1drv.com      False 2013-08-05 18:33:50  \
4       soundcloud-pax.pandora.com      False 1993-12-28 05:00:00
5         joolcomercializadora.com       True 2023-05-22 14:53:50
6        createpdf-asr.acrobat.com      False 1999-03-16 05:00:00
8                          popt.in      False 2016-05-14 16:58:55

       ctlog_earliest        ctlog_latest  ctlog_wildcard
0 2022-01-24 20:01:58 2023-06-08 20:46:06            True  \
4 2022-05-19 00:00:00 2023-06-19 23:59:59            True
5 2022-04-06 22:23:24 2023-09-22 23:59:59           False
6 2022-09-09 00:00:00 2023-10-10 23:59:59            True
8 2023-01-07 20:36:15 2023-08-15 04:16:52           False

   whois_created_dayofweek  ctlog_earliest_dayofweek
ctlog_latest_dayofweek
0                        0                         0
3  \
4                        1                         3
0
```

```
5                     0                       2
4
6                     1                       4
1
8                     5                       5
1

   domain_to_earliest_cert_delta  domain_to_latest_cert_delta
0                         3095.0                       3595.0  \
4                        10369.0                      10766.0
5                          410.0                        124.0
6                         8578.0                       8975.0
8                         2430.0                       2649.0

   whois_created_dow_sin  whois_created_dow_cos  ctlog_earliest_dow_sin
0               0.000000               1.000000                0.000000  \
4               0.918032               0.396506               -0.340712
5               0.000000               1.000000                0.728010
6               0.918032               0.396506               -0.998199
8              -0.450871               0.892589               -0.450871

   ctlog_earliest_dow_cos  ctlog_latest_dow_sin  ctlog_latest_dow_cos
0                1.000000             -0.340712             -0.940168
4               -0.940168              0.000000              1.000000
5               -0.685567             -0.998199             -0.059997
6               -0.059997              0.918032              0.396506
8                0.892589              0.918032              0.396506
       domain_to_earliest_cert_delta  whois_created_dow_sin
count                   21549.000000           21549.000000  \
mean                     3742.948397               0.140419
std                      3694.584062               0.659922
min                         0.000000              -0.998199
25%                       181.000000              -0.340712
50%                      2637.000000               0.000000
75%                      7078.000000               0.728010
max                     13445.000000               0.918032

       whois_created_dow_cos
count           21549.000000
mean                0.054288
std                 0.736128
min                -0.940168
25%                -0.685567
50%                 0.396506
75%                 0.767830
max                 1.000000
```

In [5]:

```
# convert y (malicious) to 1/0 int
y = y.astype('int')
```

```
# convert X["ctlog_wildcard"]  to 1/0 int
if "ctlog_wildcard" in X.columns:
    X["ctlog_wildcard"] = X["ctlog_wildcard"].astype('int')

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)


# random forest model

param_grid = {
    'n_estimators': [50,100,150,200],
    'max_features': ['sqrt', 'log2'],
    'max_depth' : [2,3,4,5],
    'criterion' :['gini', 'entropy']
}
```

In [6]:

```
rf = RandomForestClassifier(random_state=42)
rf_cv = GridSearchCV(estimator=rf, param_grid=param_grid, cv=5, n_jobs=-1)
rf_cv.fit(X_train, y_train.values.ravel())
```

Out[6]:

GridSearchCV

```
GridSearchCV(cv=5, estimator=RandomForestClassifier(random_state=42),
n_jobs=-1,
             param_grid={'criterion': ['gini', 'entropy'],
                         'max_depth': [2, 3, 4, 5],
                         'max_features': ['sqrt', 'log2'],
                         'n_estimators': [50, 100, 150, 200]})
```

estimator: RandomForestClassifier

```
RandomForestClassifier(random_state=42)
```

```
RandomForestClassifier
RandomForestClassifier(random_state=42)
```

In [7]:

```
bp = rf_cv.best_params_
click.echo("Best parameters set found:")
click.echo(bp)
Best parameters set found:
{'criterion': 'gini', 'max_depth': 5, 'max_features': 'sqrt',
'n_estimators': 100}
```

In [8]:

```
rf = RandomForestClassifier(random_state=42,
max_features=bp["max_features"], n_estimators=bp["n_estimators"],
max_depth=bp["max_depth"], criterion=bp["criterion"])
```

In [9]:

```
rf.fit(X_train, y_train.values.ravel())
```

Out[9]:

```
RandomForestClassifier
RandomForestClassifier(max_depth=5, random_state=42)
```

In [ ]:

In [10]:

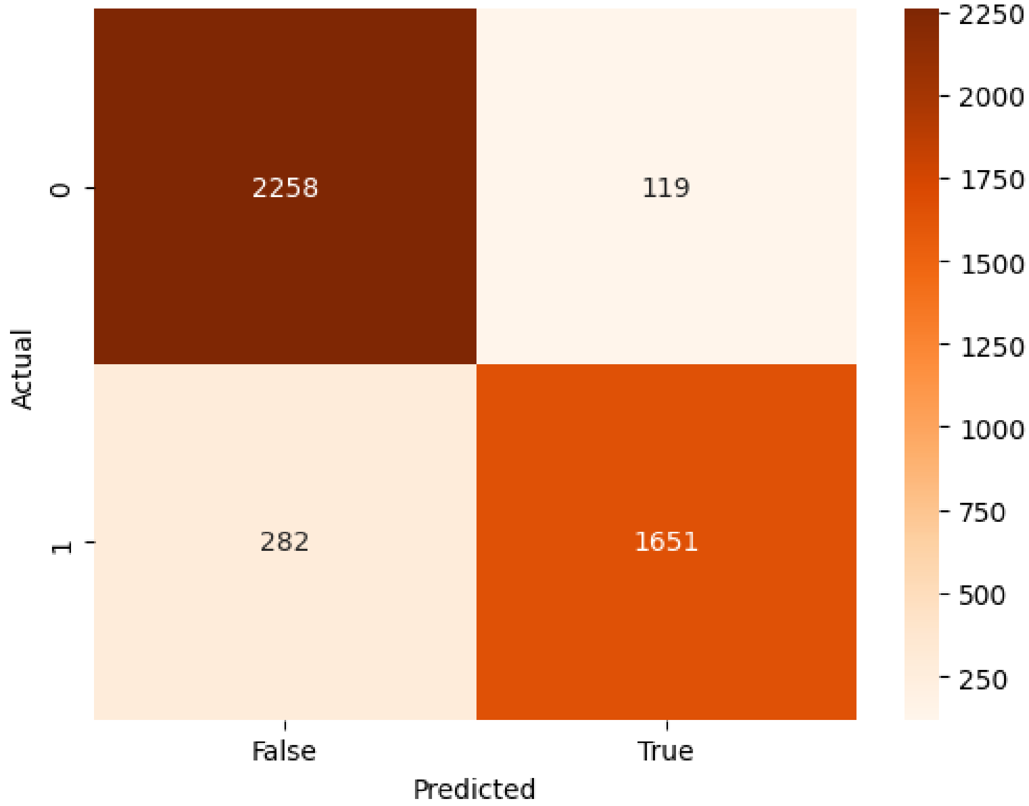
```
# Predict the malicious column using the test data
#add the incepts


y_predicted = rf.predict(X_test)

# Present the results
click.echo("Features selected:")
click.echo(X.columns)
click.echo("Confusion matrix:")
cm = confusion_matrix(y_test, y_predicted)
click.echo(cm)
click.echo("Classification report:")
click.echo(classification_report(y_test, y_predicted))


# Heatmap of confusion matrix
y_predicted

threshold = 0.5
y_predicted = [y > threshold for y in y_predicted]
data = {'Actual':    y_test.values.flatten(),
        'Predicted': y_predicted
        }

# Generate a confusion matrix and heatmap to evaluate the Type I and Type
II errors/ FP/FN etc.
df = pd.DataFrame(data, columns=['Actual','Predicted'])
cm2 = pd.crosstab(df['Actual'], df['Predicted'], rownames=['Actual'],
colnames=['Predicted'])
fig = sns.heatmap(cm2, annot=True, cmap='Oranges', fmt='g')
fig
```

```
Features selected:
Index(['domain_to_earliest_cert_delta', 'whois_created_dow_sin',
       'whois_created_dow_cos'],
      dtype='object')
Confusion matrix:
[[2258  119]
 [ 282 1651]]
Classification report:
              precision    recall  f1-score   support

           0       0.89      0.95      0.92      2377
           1       0.93      0.85      0.89      1933

    accuracy                           0.91      4310
```

```
   macro avg       0.91      0.90      0.91      4310
weighted avg       0.91      0.91      0.91      4310
```

Out[10]:

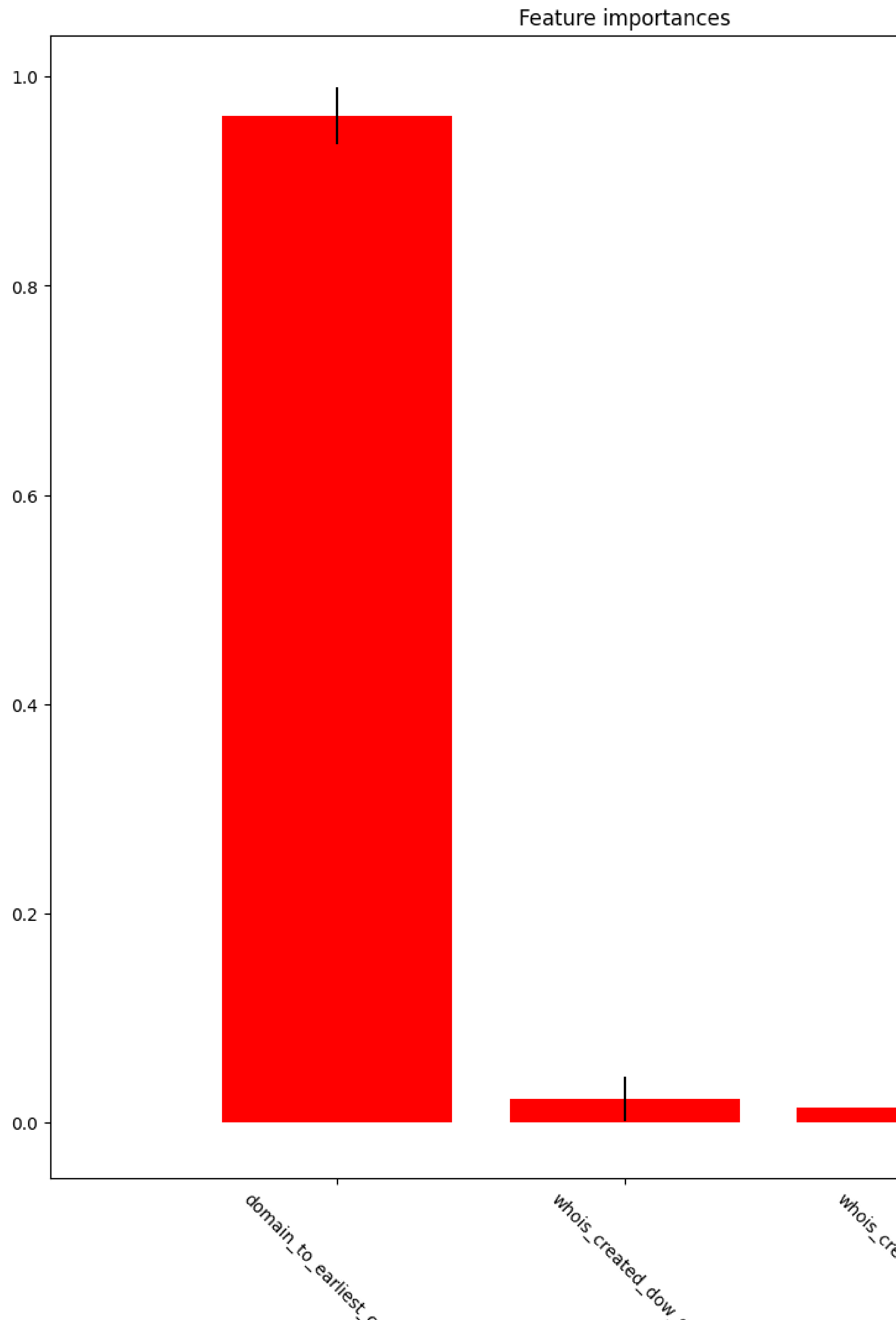
```
<Axes: xlabel='Predicted', ylabel='Actual'>
```



In [11]:

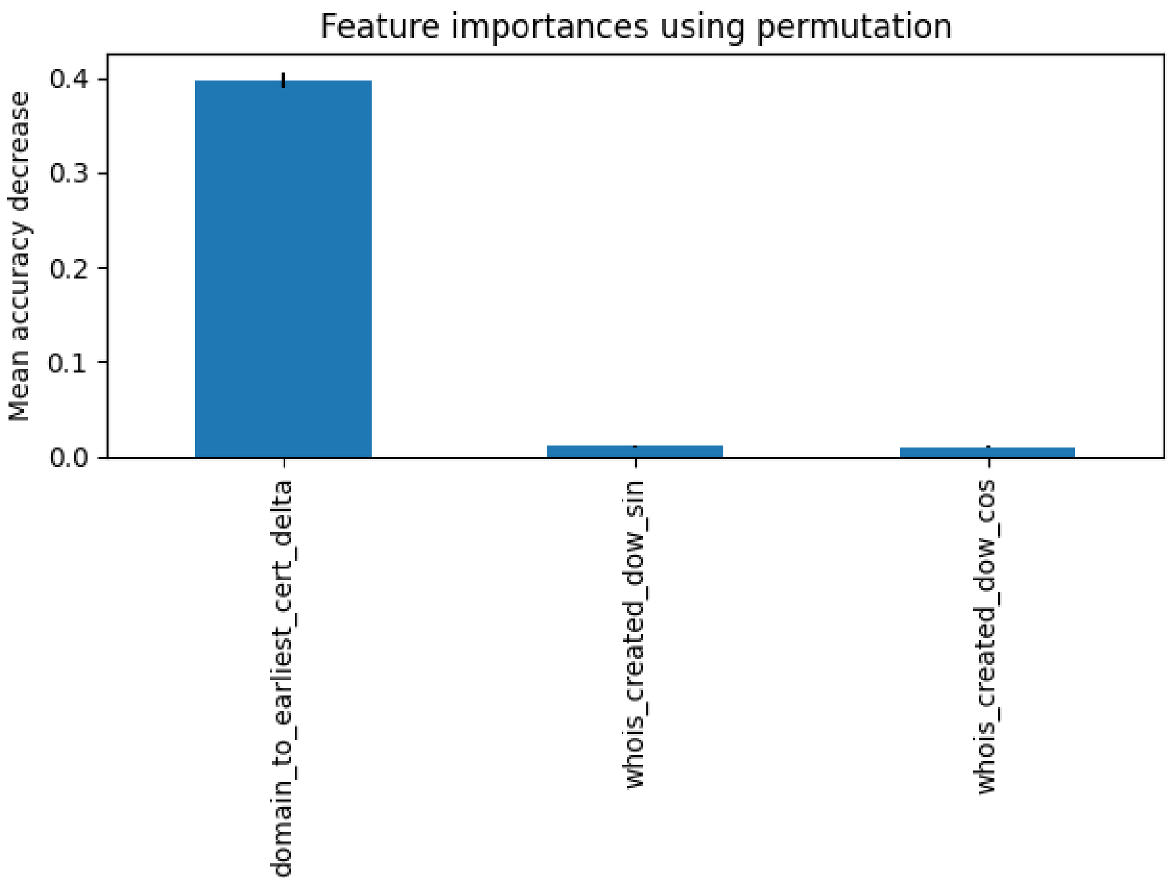
```
# plot the feature importances
importances = rf.feature_importances_
std = np.std([tree.feature_importances_ for tree in rf.estimators_],
axis=0)

indices = np.argsort(importances)[::-1]
# Print the feature ranking
click.echo("Feature ranking:")
for f in range(X.shape[1]):
    click.echo("%d. feature %s (%f)" % (f + 1, combo_features[f],
importances[indices[f]]))

# Plot the feature importances of the forest
plt.figure(figsize=(12,12))
plt.title("Feature importances")
plt.bar(range(X.shape[1]), importances[indices], color="r",
yerr=std[indices], align="center")
plt.xticks(range(X.shape[1]), X.columns[indices], rotation=-45)
plt.xlim([-1, X.shape[1]])
plt.show()
Feature ranking:
```

1. feature domain_to_earliest_cert_delta (0.962647)
2. feature whois_created_dow_sin (0.022853)
3. feature whois_created_dow_cos (0.014500)


Feature importances
1.0
0.8
0.6
0.4
0.2
0.0
domain_to_earliest_
whois_created_dow_
whois_cre

In [12]:

```
from sklearn.inspection import permutation_importance

result = permutation_importance(rf, X_test, y_test, n_repeats=100,
random_state=42, n_jobs=-1)

forest_importances = pd.Series(result.importances_mean, index=X.columns)
fig, ax = plt.subplots()
forest_importances.plot.bar(yerr=result.importances_std, ax=ax)
ax.set_title("Feature importances using permutation")
ax.set_ylabel("Mean accuracy decrease")
fig.tight_layout()
plt.show()
```



VIII. Feature Set G

In [1]:

```
import click
import pandas as pd
import glob
import os
#import sklearn
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.metrics import accuracy_score, f1_score, confusion_matrix,
classification_report, roc_auc_score, roc_curve, auc
from sklearn.preprocessing import StandardScaler
from scipy import stats
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
import statsmodels.api as sm
import matplotlib.pyplot as plt
from datetime import datetime, date
# qqplot of the data
import seaborn as sns
import math
import numpy as np
import utils

combo_features = [
    'domain_to_earliest_cert_delta',
    'ctlog_earliest_dow_sin',
    'ctlog_earliest_dow_cos',
    'ctlog_wildcard',
    'whois_created_dow_sin',
    'whois_created_dow_cos']

path = "../data/"
filename = None

epoch = datetime(1970,1,1)
# open the training data file, from ../data/ for the most recent merged
training data file saved by prepare-training-data-parallel.py
full_path = path + "merged_20230705-104357_training_adorned-engineered.csv"

# get latest file matching the glob
if not filename:
    filename = max(glob.glob(full_path), key=os.path.getctime)
    click.echo(f"Using {filename} as training data")
    df = pd.read_csv(filename, sep=",")

# randomize the rows
df = df.sample(frac=1, random_state=42).reset_index(drop=True)

click.echo(df.shape)
```

```
click.echo(df.columns)

""" # From prepare-training-data-parallel.py:
# Columns:
url
verification_time
domain
malicious
dateadded
whois_created
soa_timestamp
ctlog_earliest
ctlog_latest
ctlog_wildcard
whois_created_dayofweek,
ctlog_earliest_dayofweek,
domain_to_cert_delta
"""


# Data cleaning - there may be some epoch values in the whois_created
#  & ct_log_earliest columns, so we need to remove those rows

# convert the whois_created and ctlog_earliest, ctlog_latest columns to
datetime

df = df.loc[df["whois_created"] != ""]
df = df.loc[df["ctlog_earliest"] != ""]
df = df.loc[df["ctlog_latest"] != ""]

# some dates are have nanoseconds in them, so we need to remove the
nanosecond part
df["whois_created"] = df["whois_created"].str.split(".", n = 1, expand =
True)[0]
# some dates has additional timezone information, so we need to remove that
df["whois_created"] = df["whois_created"].apply(lambda x: " ".join(
str(x).split(" ")[:2]))


# convert the whois_created and ct_log_earliest columns to datetime
df = df.astype({"whois_created": 'datetime64[ns]', "ctlog_earliest":
'datetime64[ns]', "ctlog_latest": 'datetime64[ns]'})
df = df.loc[df["whois_created"].ne(pd.Timestamp('1970-01-01 00:00:00'))]
df = df.loc[df["ctlog_earliest"].ne(pd.Timestamp('1970-01-01 00:00:00'))]
df = df.loc[df["ctlog_latest"].ne(pd.Timestamp('1970-01-01 00:00:00'))]


# malicious is a bool
df['malicious'] = df['malicious'].astype('bool')
```

```
df['domain'] = df['domain'].astype('string')
Using ../data/merged_20230705-104357_training_adorned-engineered.csv as
training data
(35438, 13)
Index(['url', 'verification_time', 'domain', 'malicious', 'dateadded',
       'whois_created', 'soa_timestamp', 'ctlog_earliest', 'ctlog_latest',
       'ctlog_wildcard', 'whois_created_dayofweek',
'ctlog_earliest_dayofweek',
       'domain_to_cert_delta'],
      dtype='object')
```

In [2]:

```
#########
# Feature engineering
#########

# remove any rows where the whois_created or ctlog_earliest columns are
null
df = df.loc[df["whois_created"].notnull()]
df = df.loc[df["ctlog_earliest"].notnull()]

# if the data is missing the "whois_created_dayofweek" or
"ctlog_earliest_dayofweek" columns, then add them
# whois created day of the week 0 = Monday, 6 = Sunday
df["whois_created_dayofweek"] = df["whois_created"].apply(
        lambda x: x.weekday() if isinstance(x, date) else None
    )

# cert valid day of the week 0 = Monday, 6 = Sunday
df["ctlog_earliest_dayofweek"] = df["ctlog_earliest"].apply(
        lambda x: x.weekday() if isinstance(x, date) else None
    )

df["ctlog_latest_dayofweek"] = df["ctlog_latest"].apply(
        lambda x: x.weekday() if isinstance(x, date) else None
    )

# set data type of the day of week columns to int
df["whois_created_dayofweek"] =
df["whois_created_dayofweek"].astype("int64")
df["ctlog_earliest_dayofweek"] =
df["ctlog_earliest_dayofweek"].astype("int64")
df["ctlog_latest_dayofweek"] = df["ctlog_latest_dayofweek"].astype("int64")

# domain to cert delta
df["domain_to_earliest_cert_delta"] = df.apply(
        lambda x: utils.get_days_delta(
            x["ctlog_earliest"],
            x["whois_created"]
            if x["whois_created"] and x["ctlog_earliest"]
```

```
            else None,
        ),
        axis=1,
    )

# set the data type of the domain_to_cert_delta column to float
df["domain_to_earliest_cert_delta"] =
df["domain_to_earliest_cert_delta"].astype("float64")

# domain to latest cert delta
df["domain_to_latest_cert_delta"] = df.apply(
        lambda x: utils.get_days_delta(
            x["ctlog_latest"],
            x["whois_created"]
            if x["whois_created"] and x["ctlog_latest"]
            else None,
        ),
        axis=1,
    )

# set the data type of the domain_to_cert_delta column to float
df["domain_to_latest_cert_delta"] =
df["domain_to_latest_cert_delta"].astype("float64")

# drop columns we imported that we will never use
df = df.drop(columns=["url", "verification_time", "dateadded",
"soa_timestamp","domain_to_cert_delta"])

click.echo(df.head())
click.echo(df.describe(include='all'))
click.echo(df.dtypes)
                            domain  malicious       whois_created
0  i-db5p-cor001.api.p001.1drv.com      False 2013-08-05 18:33:50  \
4       soundcloud-pax.pandora.com      False 1993-12-28 05:00:00
5         joolcomercializadora.com       True 2023-05-22 14:53:50
6        createpdf-asr.acrobat.com      False 1999-03-16 05:00:00
8                          popt.in      False 2016-05-14 16:58:55

       ctlog_earliest        ctlog_latest  ctlog_wildcard
0 2022-01-24 20:01:58 2023-06-08 20:46:06            True  \
4 2022-05-19 00:00:00 2023-06-19 23:59:59            True
5 2022-04-06 22:23:24 2023-09-22 23:59:59           False
6 2022-09-09 00:00:00 2023-10-10 23:59:59            True
8 2023-01-07 20:36:15 2023-08-15 04:16:52           False

   whois_created_dayofweek  ctlog_earliest_dayofweek
ctlog_latest_dayofweek
0                        0                          0
3  \
```

```
4                        1                         3
0
5                        0                         2
4
6                        1                         4
1
8                        5                         5
1

   domain_to_earliest_cert_delta  domain_to_latest_cert_delta
0                        -3095.0                      -3595.0
4                       -10369.0                     -10766.0
5                          410.0                       -124.0
6                        -8578.0                      -8975.0
8                        -2430.0                      -2649.0
                   domain malicious                  whois_created
count               21549     21549                          21549  \
unique              21536         2                            NaN
top     www.mediafire.com     False                            NaN
freq                    2     11739                            NaN
mean                  NaN       NaN  2012-10-03 12:56:32.335050496
min                   NaN       NaN            1986-01-09 00:00:00
25%                   NaN       NaN            2003-05-25 13:35:05
50%                   NaN       NaN            2015-05-07 23:56:05
75%                   NaN       NaN            2023-03-20 15:03:16
max                   NaN       NaN            2023-07-03 08:21:24
std                   NaN       NaN                            NaN

                        ctlog_earliest                   ctlog_latest
count                            21549                          21549  \
unique                             NaN                            NaN
top                                NaN                            NaN
freq                               NaN                            NaN
mean    2022-09-26 15:45:50.943570432  2023-08-14 17:49:06.400900352
min               2021-11-30 05:24:28            2023-01-01 18:42:11
25%               2022-06-24 13:47:12            2023-07-02 08:11:07
50%               2022-10-18 21:00:14            2023-08-21 21:40:11
75%               2022-12-14 00:00:00            2023-09-21 19:41:38
max               2023-06-28 04:36:22            2023-12-31 23:59:59
std                                NaN                            NaN

       ctlog_wildcard  whois_created_dayofweek  ctlog_earliest_dayofweek
count           21549             21549.000000              21549.000000  \
unique              2                      NaN                       NaN
top             False                      NaN                       NaN
freq            13032                      NaN                       NaN
mean              NaN                 2.332823                  2.399462
min               NaN                 0.000000                  0.000000
25%               NaN                 1.000000                  1.000000
```

```
50%              NaN                 2.000000                   2.000000
75%              NaN                 4.000000                   4.000000
max              NaN                 6.000000                   6.000000
std              NaN                 1.775043                   1.897252

        ctlog_latest_dayofweek  domain_to_earliest_cert_delta
count             21549.000000                   21549.000000  \
unique                     NaN                            NaN
top                        NaN                            NaN
freq                       NaN                            NaN
mean                  2.873080                   -3645.602070
min                   0.000000                  -13445.000000
25%                   1.000000                   -7078.000000
50%                   3.000000                   -2637.000000
75%                   5.000000                      69.000000
max                   6.000000                     524.000000
std                   2.057394                    3790.677119

        domain_to_latest_cert_delta
count                  21549.000000
unique                          NaN
top                             NaN
freq                            NaN
mean                   -3967.678222
min                   -13798.000000
25%                    -7421.000000
50%                    -3009.000000
75%                     -144.000000
max                      135.000000
std                     3852.703681
domain                              string[python]
malicious                                     bool
whois_created                       datetime64[ns]
ctlog_earliest                      datetime64[ns]
ctlog_latest                        datetime64[ns]
ctlog_wildcard                                bool
whois_created_dayofweek                      int64
ctlog_earliest_dayofweek                     int64
ctlog_latest_dayofweek                       int64
domain_to_earliest_cert_delta              float64
domain_to_latest_cert_delta                float64
dtype: object
```

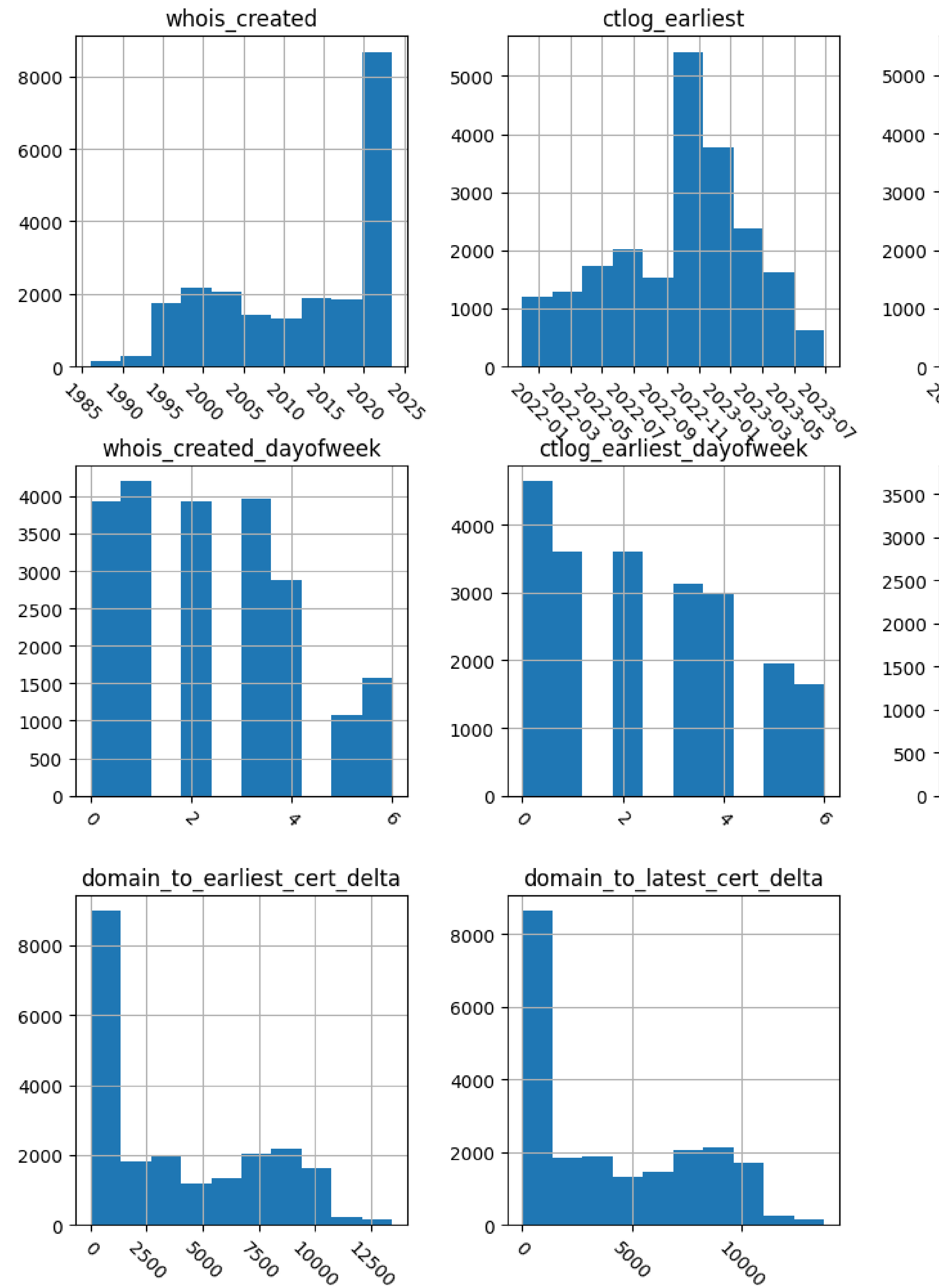
In [3]:

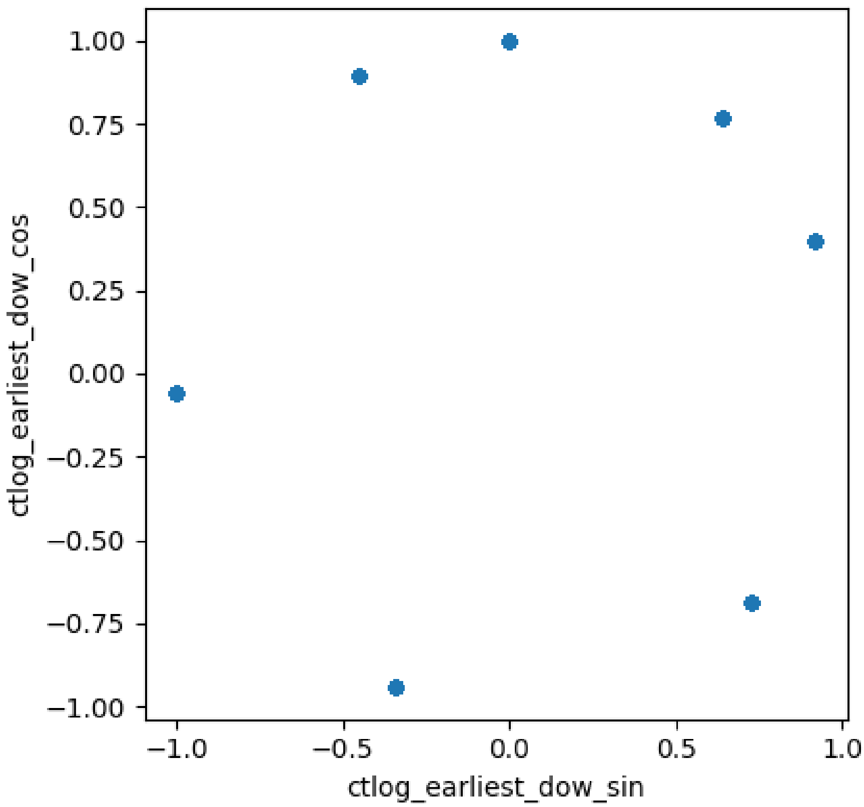
```
# absolute value of the domain_to_cert_delta
df["domain_to_earliest_cert_delta"] =
abs(df["domain_to_earliest_cert_delta"])
df["domain_to_latest_cert_delta"] = abs(df["domain_to_latest_cert_delta"])

df.hist(figsize=(12,12), xrot=-45)
```

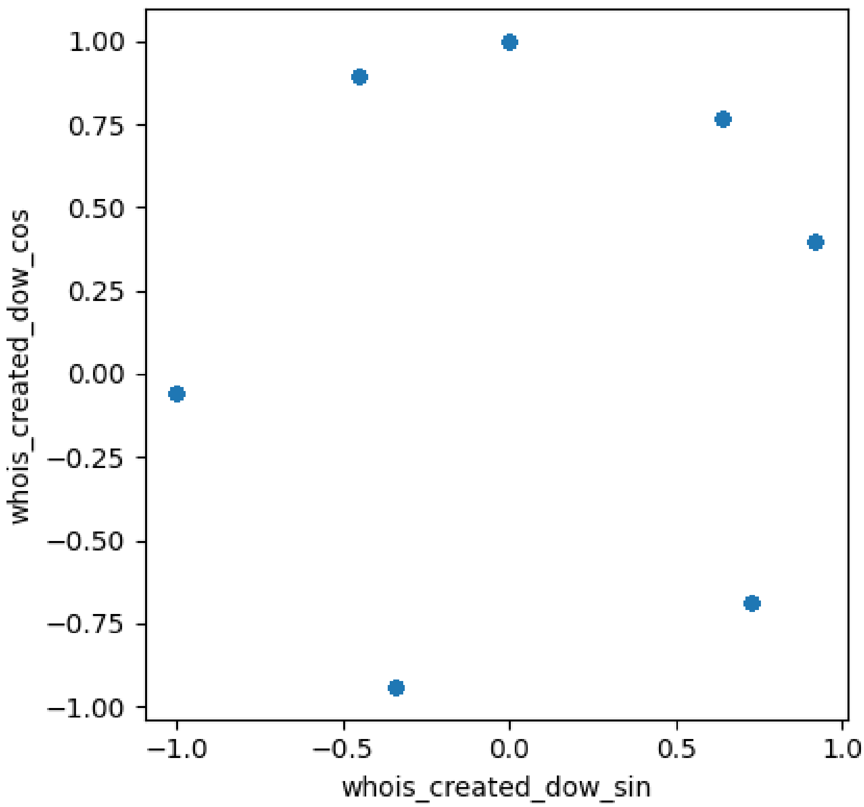
```
col_index = df.columns.get_loc("whois_created_dayofweek")
df["whois_created_dow_sin"] = df.apply(lambda row:
math.sin(360/7*(row[col_index])),axis=1)
df["whois_created_dow_cos"] = df.apply(lambda row:
math.cos(360/7*(row[col_index])),axis=1)

col_index = df.columns.get_loc("ctlog_earliest_dayofweek")
df["ctlog_earliest_dow_sin"] = df.apply(lambda row:
math.sin(360/7*(row[col_index])),axis=1)
df["ctlog_earliest_dow_cos"] = df.apply(lambda row:
math.cos(360/7*(row[col_index])),axis=1)

col_index = df.columns.get_loc("ctlog_latest_dayofweek")
df["ctlog_latest_dow_sin"] = df.apply(lambda row:
math.sin(360/7*(row[col_index])),axis=1)
df["ctlog_latest_dow_cos"] = df.apply(lambda row:
math.cos(360/7*(row[col_index])),axis=1)


df.plot.scatter(x="ctlog_earliest_dow_sin",
y="ctlog_earliest_dow_cos").set_aspect('equal')
df.plot.scatter(x="whois_created_dow_sin",
y="whois_created_dow_cos").set_aspect('equal')
df.plot.scatter(x="ctlog_latest_dow_sin",
y="ctlog_latest_dow_cos").set_aspect('equal')

"""
# add one hot encode ctlog_earliest_not_before_dayofweek
df = pd.get_dummies(
        df,
        columns=["ctlog_earliest_dayofweek"],
        prefix=["ctlog_earliest_dow"],
    )
# add one hot encode whois_created_dayofweek to columns
df = pd.get_dummies(
        df, columns=["whois_created_dayofweek"], prefix="whois_dow"
    )
    """

# Summary statistics
click.echo(df.describe(include='all'))
                   domain malicious                    whois_created
count               21549     21549                            21549  \
unique              21536         2                              NaN
top      www.mediafire.com     False                              NaN
freq                    2     11739                              NaN
mean                  NaN       NaN  2012-10-03 12:56:32.335050496
min                   NaN       NaN            1986-01-09 00:00:00
```

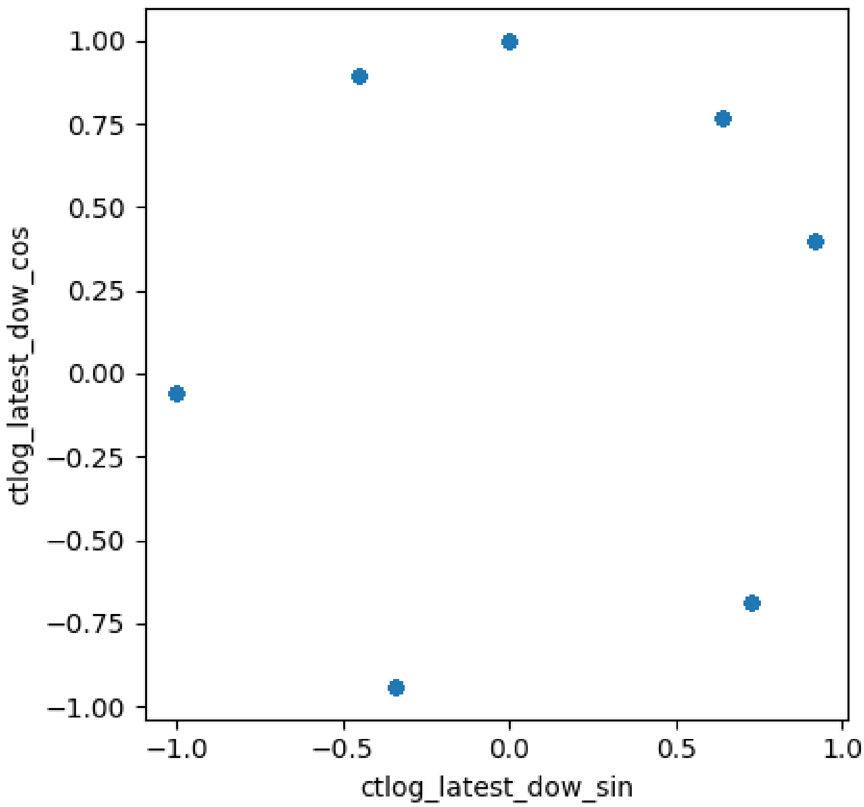
```
25%                    NaN       NaN            2003-05-25 13:35:05
50%                    NaN       NaN            2015-05-07 23:56:05
75%                    NaN       NaN            2023-03-20 15:03:16
max                    NaN       NaN            2023-07-03 08:21:24
std                    NaN       NaN                            NaN

                       ctlog_earliest                   ctlog_latest
count                           21549                          21549  \
unique                            NaN                            NaN
top                               NaN                            NaN
freq                              NaN                            NaN
mean    2022-09-26 15:45:50.943570432  2023-08-14 17:49:06.400900352
min               2021-11-30 05:24:28            2023-01-01 18:42:11
25%               2022-06-24 13:47:12            2023-07-02 08:11:07
50%               2022-10-18 21:00:14            2023-08-21 21:40:11
75%               2022-12-14 00:00:00            2023-09-21 19:41:38
max               2023-06-28 04:36:22            2023-12-31 23:59:59
std                               NaN                            NaN

       ctlog_wildcard  whois_created_dayofweek  ctlog_earliest_dayofweek
count           21549             21549.000000              21549.000000  \
unique              2                      NaN                       NaN
top             False                      NaN                       NaN
freq            13032                      NaN                       NaN
mean              NaN                 2.332823                  2.399462
min               NaN                 0.000000                  0.000000
25%               NaN                 1.000000                  1.000000
50%               NaN                 2.000000                  2.000000
75%               NaN                 4.000000                  4.000000
max               NaN                 6.000000                  6.000000
std               NaN                 1.775043                  1.897252

        ctlog_latest_dayofweek  domain_to_earliest_cert_delta
count             21549.000000                   21549.000000  \
unique                     NaN                            NaN
top                        NaN                            NaN
freq                       NaN                            NaN
mean                  2.873080                    3742.948397
min                   0.000000                       0.000000
25%                   1.000000                     181.000000
50%                   3.000000                    2637.000000
75%                   5.000000                    7078.000000
max                   6.000000                   13445.000000
std                   2.057394                    3694.584062

        domain_to_latest_cert_delta  whois_created_dow_sin
count                  21549.000000           21549.000000  \
unique                          NaN                    NaN
top                             NaN                    NaN
```

```
freq                       NaN                      NaN
mean               3969.491206                 0.140419
min                   0.000000                -0.998199
25%                 144.000000                -0.340712
50%                3009.000000                 0.000000
75%                7421.000000                 0.728010
max               13798.000000                 0.918032
std                3850.835626                 0.659922

        whois_created_dow_cos  ctlog_earliest_dow_sin
ctlog_earliest_dow_cos
count            21549.000000            21549.000000
21549.000000  \
unique                    NaN                     NaN
NaN
top                       NaN                     NaN
NaN
freq                      NaN                     NaN
NaN
mean                 0.054288                0.095357
0.161451
min                 -0.940168               -0.998199               -
0.940168
25%                 -0.685567               -0.340712               -
0.685567
50%                  0.396506                0.000000
0.396506
75%                  0.767830                0.728010
0.892589
max                  1.000000                0.918032
1.000000
std                  0.736128                0.651782
0.734891

        ctlog_latest_dow_sin  ctlog_latest_dow_cos
count           21549.000000          21549.000000
unique                   NaN                   NaN
top                      NaN                   NaN
freq                     NaN                   NaN
mean                0.096253              0.255578
min                -0.998199             -0.940168
25%                -0.450871             -0.685567
50%                 0.000000              0.396506
75%                 0.728010              0.892589
max                 0.918032              1.000000
std                 0.651597              0.707728
```


whois_created
0
2000
4000
6000
8000
1985
1990
1995
2000
2005
2010
2015
2020
2025
ctlog_earliest
0
1000
2000
3000
4000
5000
2022-01
2022-03
2022-05
2022-07
2022-09
2022-11
2023-01
2023-03
2023-05
2023-07
whois_created_dayofweek
0
500
1000
1500
2000
2500
3000
3500
4000
0
2
4
6
ctlog_earliest_dayofweek
0
1000
2000
3000
4000
0
2
4
6
domain_to_earliest_cert_delta
0
2000
4000
6000
8000
0
2500
5000
7500
10000
12500
domain_to_latest_cert_delta
0
2000
4000
6000
8000
0
5000
10000


ctlog_earliest_dow_cos
ctlog_earliest_dow_sin
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−1.0
−0.5
0.0
0.5
1.0

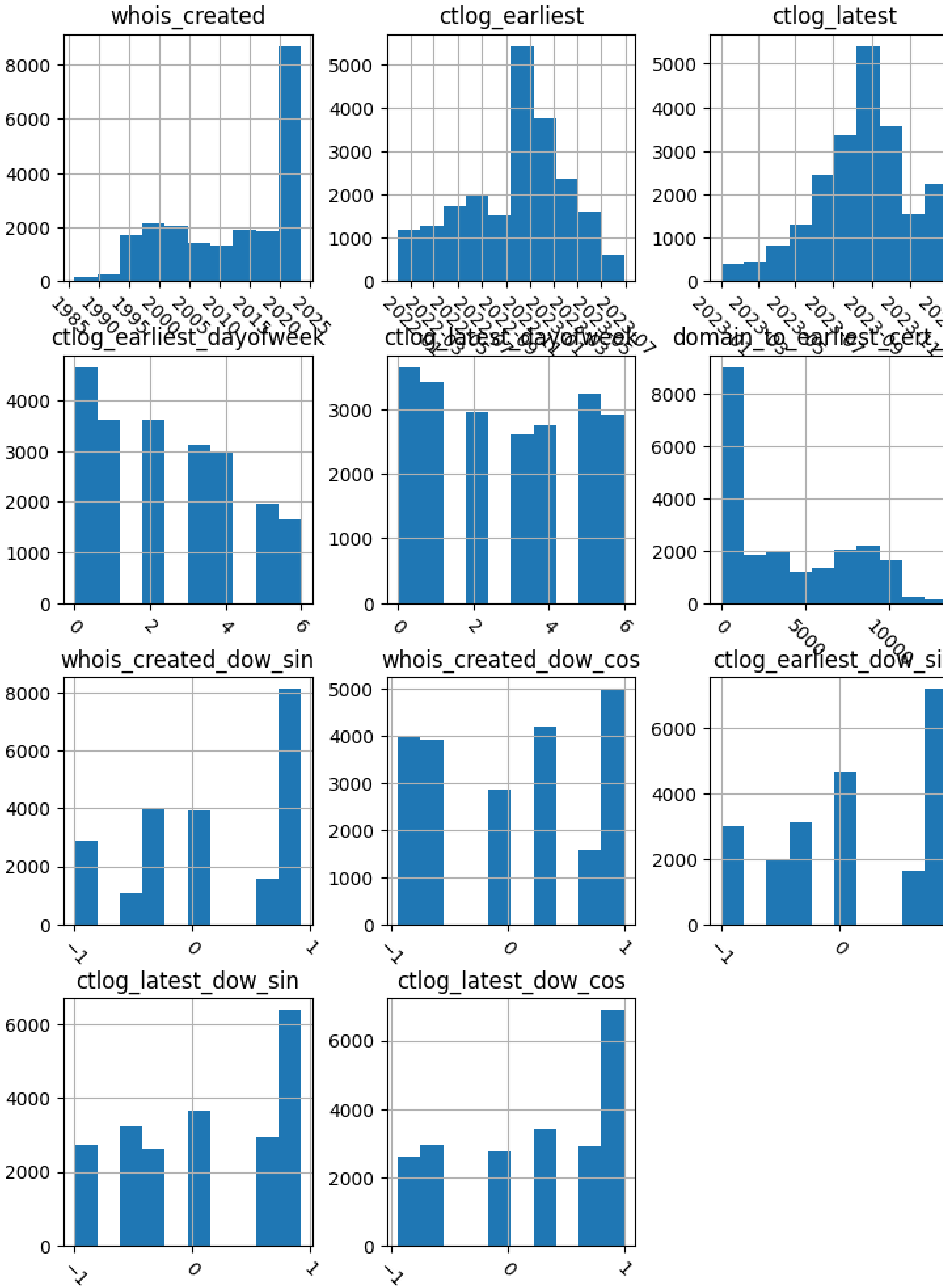

whois_created_dow_cos
whois_created_dow_sin
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−1.0
−0.5
0.0
0.5
1.0

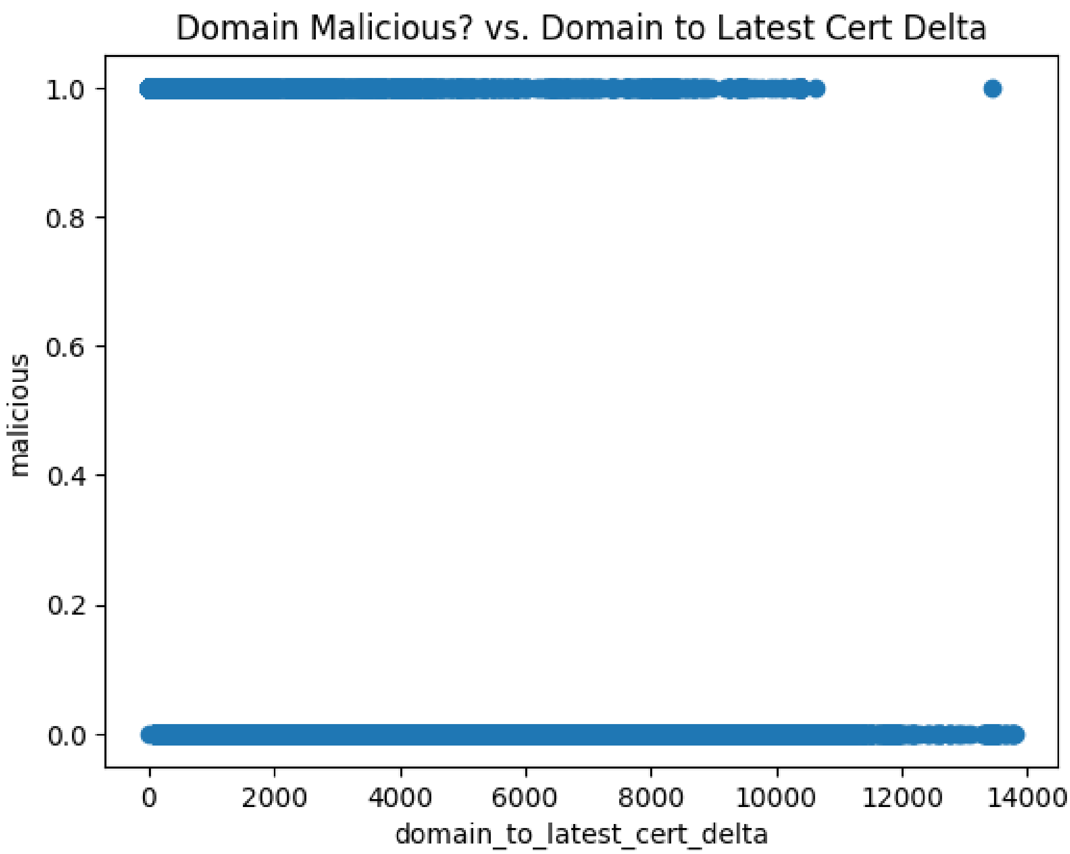


In [4]:

```
# Plot histograms
df.hist(figsize=(12,12), xrot=-45)

# Create a scatter plot of the data, showing malicious and benign domains
# plot the data as a scatter plot
plt.scatter(df["domain_to_earliest_cert_delta"], df["malicious"])
plt.xlabel("domain_to_earliest_cert_delta")
plt.ylabel("malicious")
plt.title("Domain Malicious? vs. Domain to Earliest Cert Delta")
plt.show()
# and for the latest
plt.scatter(df["domain_to_latest_cert_delta"], df["malicious"])
plt.xlabel("domain_to_latest_cert_delta")
plt.ylabel("malicious")
plt.title("Domain Malicious? vs. Domain to Latest Cert Delta")
plt.show()



click.echo(df.head())

# print a count of malicious and benign values
click.echo(df["malicious"].value_counts())
# deduplicate any rows, there shouldn't be any, but just in case
df = df.drop_duplicates()
click.echo(df["malicious"].value_counts())
```

```
click.echo(df.head())

X = df.drop(["malicious","domain", "ctlog_earliest", "ctlog_latest",
"whois_created", "whois_created_dayofweek", "ctlog_earliest_dayofweek"],
axis=1)
X = df.filter(combo_features)
y = df.filter(["malicious"])


click.echo(X.describe())
```


whois_created
ctlog_earliest
ctlog_latest
ctlog_earliest_dayofweek
ctlog_latest_dayofweek
whois_created_dow_sin
whois_created_dow_cos
ctlog_latest_dow_sin
ctlog_latest_dow_cos




```
                              domain  malicious       whois_created
0  i-db5p-cor001.api.p001.1drv.com      False 2013-08-05 18:33:50   \
4       soundcloud-pax.pandora.com      False 1993-12-28 05:00:00
5         joolcomercializadora.com       True 2023-05-22 14:53:50
6        createpdf-asr.acrobat.com      False 1999-03-16 05:00:00
8                          popt.in      False 2016-05-14 16:58:55

        ctlog_earliest        ctlog_latest  ctlog_wildcard  
0 2022-01-24 20:01:58 2023-06-08 20:46:06            True   \
4 2022-05-19 00:00:00 2023-06-19 23:59:59            True
5 2022-04-06 22:23:24 2023-09-22 23:59:59           False
6 2022-09-09 00:00:00 2023-10-10 23:59:59            True
8 2023-01-07 20:36:15 2023-08-15 04:16:52           False

   whois_created_dayofweek  ctlog_earliest_dayofweek
ctlog_latest_dayofweek
0                        0                          0
3  \
4                        1                          3
0
5                        0                          2
4
6                        1                          4
1
8                        5                          5
1
```

```
   domain_to_earliest_cert_delta  domain_to_latest_cert_delta
0                         3095.0                       3595.0  \
4                        10369.0                      10766.0
5                          410.0                        124.0
6                         8578.0                       8975.0
8                         2430.0                       2649.0

   whois_created_dow_sin  whois_created_dow_cos  ctlog_earliest_dow_sin
0               0.000000               1.000000                0.000000  \
4               0.918032               0.396506               -0.340712
5               0.000000               1.000000                0.728010
6               0.918032               0.396506               -0.998199
8              -0.450871               0.892589               -0.450871

   ctlog_earliest_dow_cos  ctlog_latest_dow_sin  ctlog_latest_dow_cos
0                1.000000             -0.340712             -0.940168
4               -0.940168              0.000000              1.000000
5               -0.685567             -0.998199             -0.059997
6               -0.059997              0.918032              0.396506
8                0.892589              0.918032              0.396506
malicious
False    11739
True      9810
Name: count, dtype: int64
malicious
False    11739
True      9810
Name: count, dtype: int64
                            domain  malicious       whois_created
0  i-db5p-cor001.api.p001.1drv.com      False 2013-08-05 18:33:50  \
4       soundcloud-pax.pandora.com      False 1993-12-28 05:00:00
5         joolcomercializadora.com       True 2023-05-22 14:53:50
6        createpdf-asr.acrobat.com      False 1999-03-16 05:00:00
8                          popt.in      False 2016-05-14 16:58:55

       ctlog_earliest        ctlog_latest  ctlog_wildcard
0 2022-01-24 20:01:58 2023-06-08 20:46:06            True  \
4 2022-05-19 00:00:00 2023-06-19 23:59:59            True
5 2022-04-06 22:23:24 2023-09-22 23:59:59           False
6 2022-09-09 00:00:00 2023-10-10 23:59:59            True
8 2023-01-07 20:36:15 2023-08-15 04:16:52           False

   whois_created_dayofweek  ctlog_earliest_dayofweek
ctlog_latest_dayofweek
0                        0                         0
3  \
4                        1                         3
0
```

```
5                        0                         2
4
6                        1                         4
1
8                        5                         5
1

   domain_to_earliest_cert_delta  domain_to_latest_cert_delta
0                         3095.0                       3595.0  \
4                        10369.0                      10766.0
5                          410.0                        124.0
6                         8578.0                       8975.0
8                         2430.0                       2649.0

   whois_created_dow_sin  whois_created_dow_cos  ctlog_earliest_dow_sin
0               0.000000               1.000000                0.000000  \
4               0.918032               0.396506               -0.340712
5               0.000000               1.000000                0.728010
6               0.918032               0.396506               -0.998199
8              -0.450871               0.892589               -0.450871

   ctlog_earliest_dow_cos  ctlog_latest_dow_sin  ctlog_latest_dow_cos
0                1.000000             -0.340712             -0.940168
4               -0.940168              0.000000              1.000000
5               -0.685567             -0.998199             -0.059997
6               -0.059997              0.918032              0.396506
8                0.892589              0.918032              0.396506
       domain_to_earliest_cert_delta  ctlog_earliest_dow_sin
count                   21549.000000            21549.000000  \
mean                     3742.948397                0.095357
std                      3694.584062                0.651782
min                         0.000000               -0.998199
25%                       181.000000               -0.340712
50%                      2637.000000                0.000000
75%                      7078.000000                0.728010
max                     13445.000000                0.918032

       ctlog_earliest_dow_cos  whois_created_dow_sin  whois_created_dow_cos
count            21549.000000           21549.000000           21549.000000
mean                 0.161451               0.140419               0.054288
std                  0.734891               0.659922               0.736128
min                 -0.940168              -0.998199              -0.940168
25%                 -0.685567              -0.340712              -0.685567
50%                  0.396506               0.000000               0.396506
75%                  0.892589               0.728010               0.767830
max                  1.000000               0.918032               1.000000
```

In [5]:

```
# convert y (malicious) to 1/0 int
y = y.astype('int')
```

```
# convert X["ctlog_wildcard"]  to 1/0 int
if "ctlog_wildcard" in X.columns:
    X["ctlog_wildcard"] = X["ctlog_wildcard"].astype('int')

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)


# random forest model

param_grid = {
    'n_estimators': [50,100,150,200],
    'max_features': ['sqrt', 'log2'],
    'max_depth' : [2,3,4,5],
    'criterion' :['gini', 'entropy']
}
```

In [6]:

```
rf = RandomForestClassifier(random_state=42)
rf_cv = GridSearchCV(estimator=rf, param_grid=param_grid, cv=5, n_jobs=-1)
rf_cv.fit(X_train, y_train.values.ravel())
```

Out[6]:

GridSearchCV

```
GridSearchCV(cv=5, estimator=RandomForestClassifier(random_state=42),
n_jobs=-1,
             param_grid={'criterion': ['gini', 'entropy'],
                         'max_depth': [2, 3, 4, 5],
                         'max_features': ['sqrt', 'log2'],
                         'n_estimators': [50, 100, 150, 200]})
```

estimator: RandomForestClassifier

```
RandomForestClassifier(random_state=42)
```

RandomForestClassifier

```
RandomForestClassifier(random_state=42)
```

In [7]:

```
bp = rf_cv.best_params_
click.echo("Best parameters set found:")
click.echo(bp)
Best parameters set found:
{'criterion': 'gini', 'max_depth': 5, 'max_features': 'sqrt',
'n_estimators': 150}
```

In [8]:

```
rf = RandomForestClassifier(random_state=42,
max_features=bp["max_features"], n_estimators=bp["n_estimators"],
max_depth=bp["max_depth"], criterion=bp["criterion"])
```

In [9]:

```
rf.fit(X_train, y_train.values.ravel())
```

Out[9]:

RandomForestClassifier

```
RandomForestClassifier(max_depth=5, n_estimators=150, random_state=42)
```

In [ ]:

In [10]:

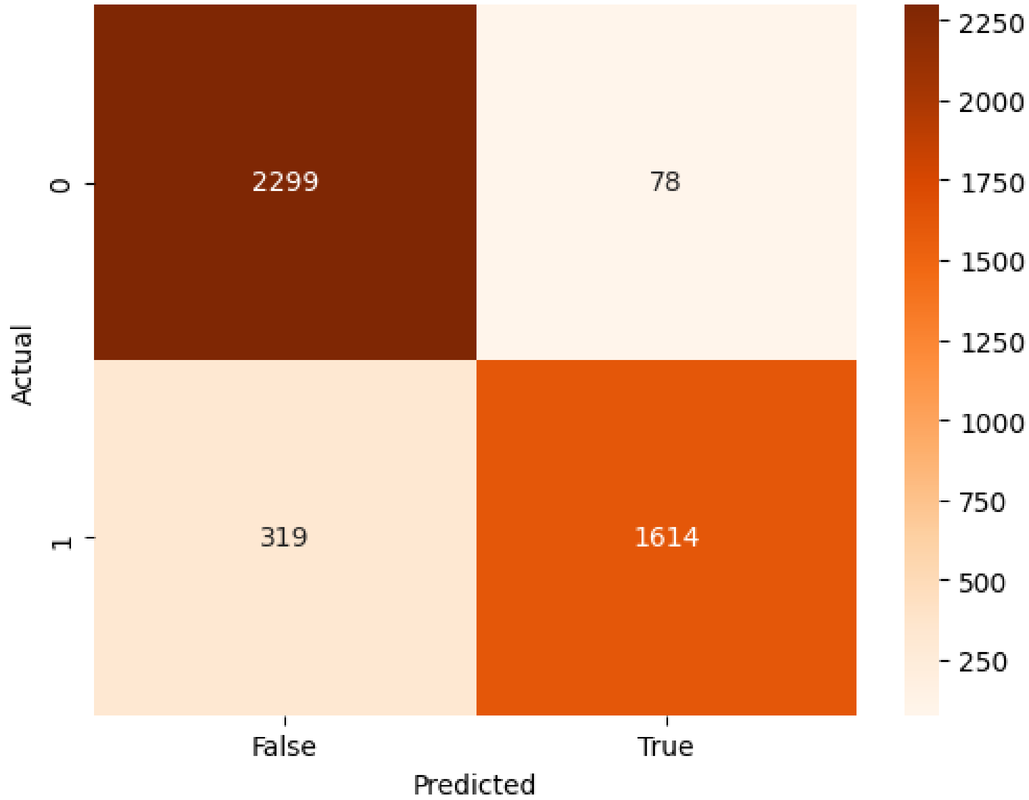
```
# Predict the malicious column using the test data
#add the incepts


y_predicted = rf.predict(X_test)

# Present the results
click.echo("Features selected:")
click.echo(X.columns)
click.echo("Confusion matrix:")
cm = confusion_matrix(y_test, y_predicted)
click.echo(cm)
click.echo("Classification report:")
click.echo(classification_report(y_test, y_predicted))


# Heatmap of confusion matrix
y_predicted

threshold = 0.5
y_predicted = [y > threshold for y in y_predicted]
data = {'Actual':    y_test.values.flatten(),
        'Predicted': y_predicted
        }

# Generate a confusion matrix and heatmap to evaluate the Type I and Type
II errors/ FP/FN etc.
df = pd.DataFrame(data, columns=['Actual','Predicted'])
cm2 = pd.crosstab(df['Actual'], df['Predicted'], rownames=['Actual'],
colnames=['Predicted'])
fig = sns.heatmap(cm2, annot=True, cmap='Oranges', fmt='g')
fig
```

```
Features selected:
Index(['domain_to_earliest_cert_delta', 'ctlog_earliest_dow_sin',
       'ctlog_earliest_dow_cos', 'ctlog_wildcard', 'whois_created_dow_sin',
       'whois_created_dow_cos'],
      dtype='object')
Confusion matrix:
[[2299   78]
 [ 319 1614]]
Classification report:
              precision    recall  f1-score   support

           0       0.88      0.97      0.92      2377
           1       0.95      0.83      0.89      1933
```

```
    accuracy                           0.91      4310
   macro avg       0.92      0.90      0.91      4310
weighted avg       0.91      0.91      0.91      4310
```

Out[10]:

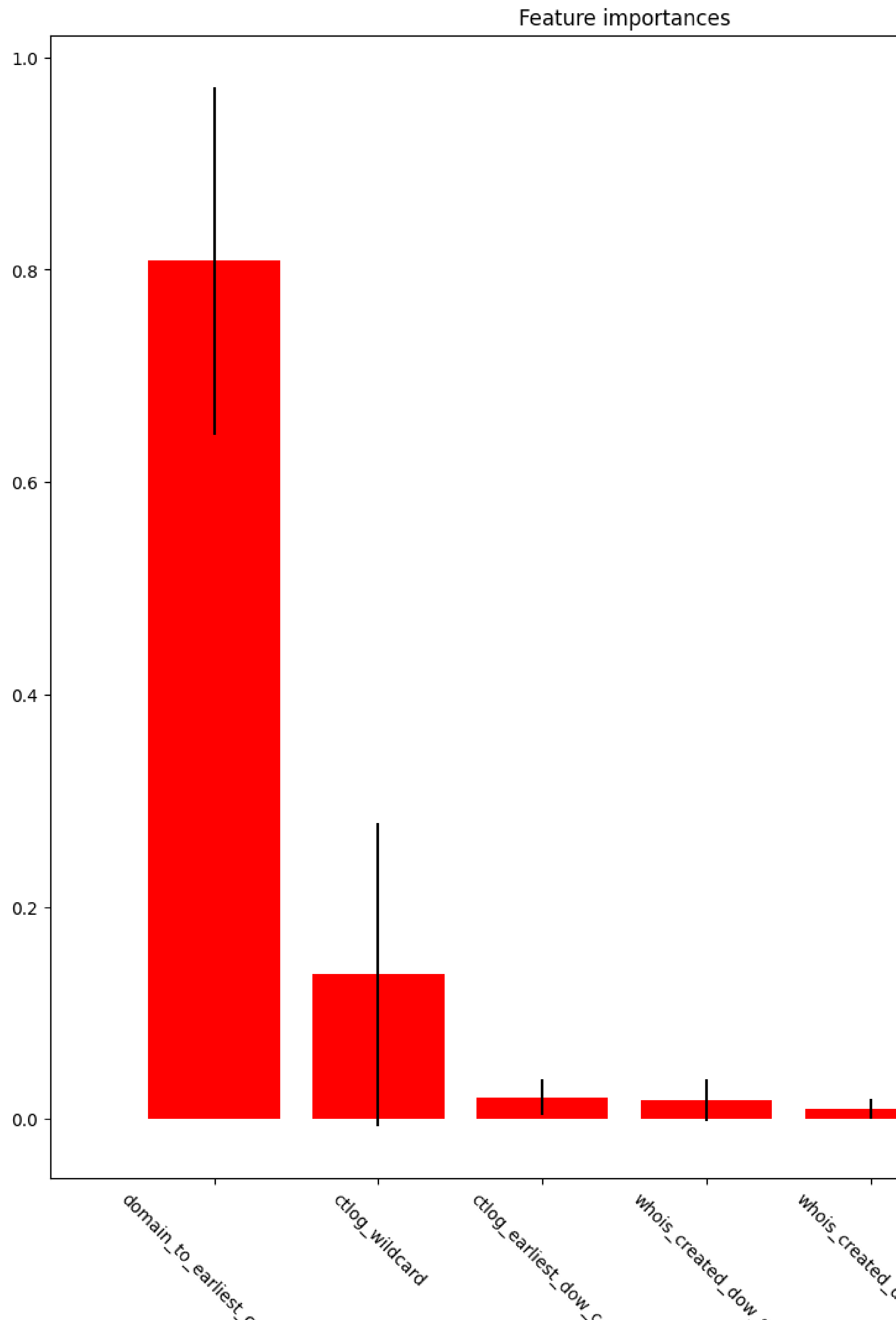
```
<Axes: xlabel='Predicted', ylabel='Actual'>
```



In [11]:

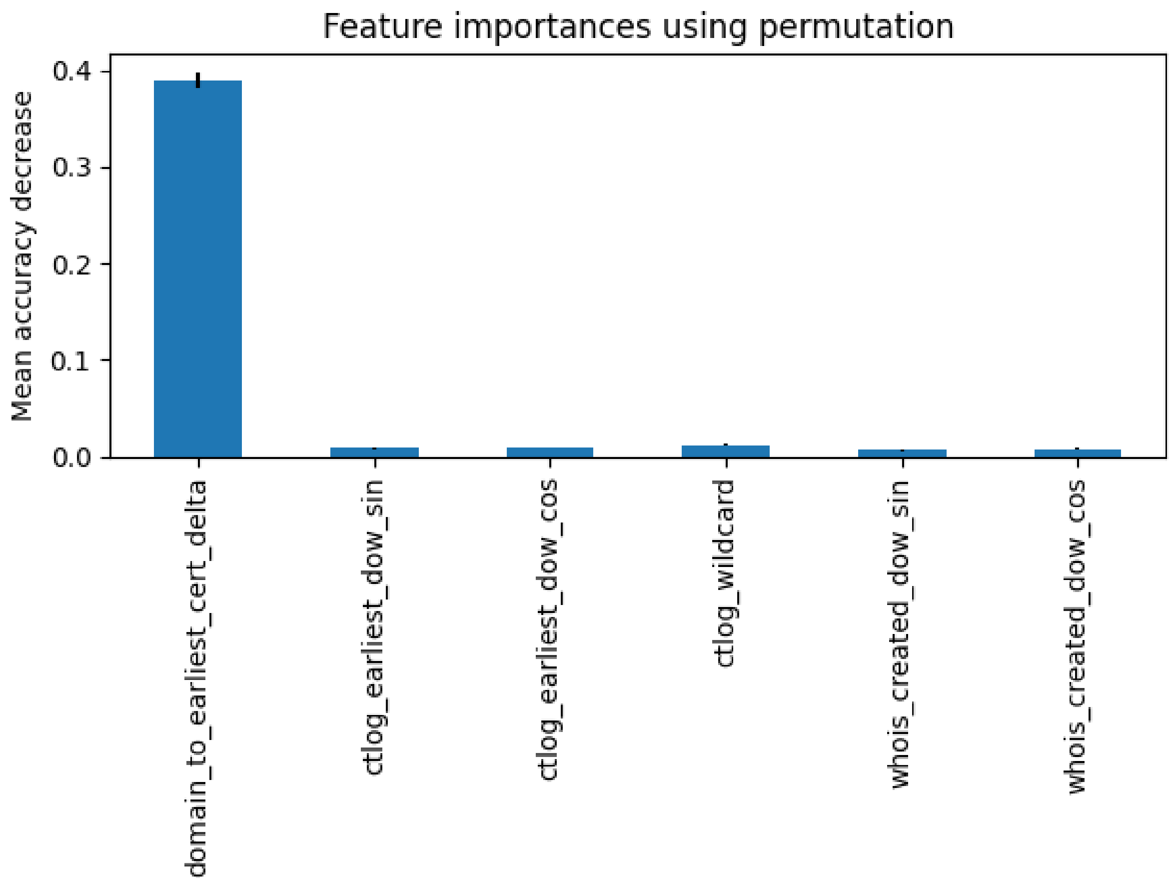
```
# plot the feature importances
importances = rf.feature_importances_
std = np.std([tree.feature_importances_ for tree in rf.estimators_],
axis=0)

indices = np.argsort(importances)[::-1]
# Print the feature ranking
click.echo("Feature ranking:")
for f in range(X.shape[1]):
    click.echo("%d. feature %s (%f)" % (f + 1, combo_features[indices[f]],
importances[indices[f]]))


# Plot the feature importances of the forest
plt.figure(figsize=(12,12))
plt.title("Feature importances")
plt.bar(range(X.shape[1]), importances[indices], color="r",
yerr=std[indices], align="center")
plt.xticks(range(X.shape[1]), X.columns[indices], rotation=-45)
plt.xlim([-1, X.shape[1]])
```

```
plt.show()
Feature ranking:
1. feature domain_to_earliest_cert_delta (0.808437)
2. feature ctlog_wildcard (0.136489)
3. feature ctlog_earliest_dow_cos (0.020678)
4. feature whois_created_dow_sin (0.017815)
5. feature whois_created_dow_cos (0.009463)
6. feature ctlog_earliest_dow_sin (0.007118)
```


Feature importances
1.0
0.8
0.6
0.4
0.2
0.0
ctlog_wildcard

In [12]:

```
from sklearn.inspection import permutation_importance

result = permutation_importance(rf, X_test, y_test, n_repeats=100,
random_state=42, n_jobs=-1)

forest_importances = pd.Series(result.importances_mean, index=X.columns)
fig, ax = plt.subplots()
forest_importances.plot.bar(yerr=result.importances_std, ax=ax)
ax.set_title("Feature importances using permutation")
ax.set_ylabel("Mean accuracy decrease")
fig.tight_layout()
plt.show()
```



IX. Feature Set H

In [1]:

```
import click
import pandas as pd
import glob
import os
#import sklearn
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.metrics import accuracy_score, f1_score, confusion_matrix,
classification_report, roc_auc_score, roc_curve, auc
from sklearn.preprocessing import StandardScaler
from scipy import stats
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
import matplotlib.pyplot as plt
from datetime import datetime, date
# qqplot of the data
import seaborn as sns
import math
import numpy as np
import utils

combo_features =[
    'domain_to_earliest_cert_delta',
    'ctlog_earliest_dow_sin',
    'ctlog_earliest_dow_cos',
    'ctlog_wildcard',
    'whois_created_dow_sin',
    'whois_created_dow_cos',
    'domain_to_latest_cert_delta',
    'ctlog_latest_dow_sin',
    'ctlog_latest_dow_cos'
    ]

path = "../data/"
filename = None

epoch = datetime(1970,1,1)
# open the training data file, from ../data/ for the most recent merged
training data file saved by prepare-training-data-parallel.py
full_path = path + "merged_20230705-104357_training_adorned-engineered.csv"

# get latest file matching the glob
if not filename:
    filename = max(glob.glob(full_path), key=os.path.getctime)
    click.echo(f"Using {filename} as training data")
    df = pd.read_csv(filename, sep=",")

# randomize the rows
```

```
df = df.sample(frac=1, random_state=42).reset_index(drop=True)

click.echo(df.shape)
click.echo(df.columns)

""" # From prepare-training-data-parallel.py:
# Columns:
url
verification_time
domain
malicious
dateadded
whois_created
soa_timestamp
ctlog_earliest
ctlog_latest
ctlog_wildcard
whois_created_dayofweek,
ctlog_earliest_dayofweek,
domain_to_cert_delta
"""


# Data cleaning - there may be some epoch values in the whois_created
#  & ct_log_earliest columns, so we need to remove those rows

# convert the whois_created and ctlog_earliest, ctlog_latest columns to
datetime

df = df.loc[df["whois_created"] != ""]
df = df.loc[df["ctlog_earliest"] != ""]
df = df.loc[df["ctlog_latest"] != ""]

# some dates are have nanoseconds in them, so we need to remove the
nanosecond part
df["whois_created"] = df["whois_created"].str.split(".", n = 1, expand =
True)[0]
# some dates has additional timezone information, so we need to remove that
df["whois_created"] = df["whois_created"].apply(lambda x: " ".join(
str(x).split(" ")[:2]))


# convert the whois_created and ct_log_earliest columns to datetime
df = df.astype({"whois_created": 'datetime64[ns]', "ctlog_earliest":
'datetime64[ns]', "ctlog_latest": 'datetime64[ns]'})
df = df.loc[df["whois_created"].ne(pd.Timestamp('1970-01-01 00:00:00'))]
df = df.loc[df["ctlog_earliest"].ne(pd.Timestamp('1970-01-01 00:00:00'))]
df = df.loc[df["ctlog_latest"].ne(pd.Timestamp('1970-01-01 00:00:00'))]
```

```
# malicious is a bool
df['malicious'] = df['malicious'].astype('bool')
df['domain'] = df['domain'].astype('string')
Using ../data/merged_20230705-104357_training_adorned-engineered.csv as
training data
(35438, 13)
Index(['url', 'verification_time', 'domain', 'malicious', 'dateadded',
       'whois_created', 'soa_timestamp', 'ctlog_earliest', 'ctlog_latest',
       'ctlog_wildcard', 'whois_created_dayofweek',
'ctlog_earliest_dayofweek',
       'domain_to_cert_delta'],
      dtype='object')
```

In [ ]:

In [2]:

```
#########
# Feature engineering
#########

# remove any rows where the whois_created or ctlog_earliest columns are
null
df = df.loc[df["whois_created"].notnull()]
df = df.loc[df["ctlog_earliest"].notnull()]

# if the data is missing the "whois_created_dayofweek" or
"ctlog_earliest_dayofweek" columns, then add them
# whois created day of the week 0 = Monday, 6 = Sunday
df["whois_created_dayofweek"] = df["whois_created"].apply(
        lambda x: x.weekday() if isinstance(x, date) else None
    )

# cert valid day of the week 0 = Monday, 6 = Sunday
df["ctlog_earliest_dayofweek"] = df["ctlog_earliest"].apply(
        lambda x: x.weekday() if isinstance(x, date) else None
    )

df["ctlog_latest_dayofweek"] = df["ctlog_latest"].apply(
        lambda x: x.weekday() if isinstance(x, date) else None
    )

# set data type of the day of week columns to int
df["whois_created_dayofweek"] =
df["whois_created_dayofweek"].astype("int64")
df["ctlog_earliest_dayofweek"] =
df["ctlog_earliest_dayofweek"].astype("int64")
df["ctlog_latest_dayofweek"] = df["ctlog_latest_dayofweek"].astype("int64")

# domain to cert delta
```

```
df["domain_to_earliest_cert_delta"] = df.apply(
        lambda x: utils.get_days_delta(
            x["ctlog_earliest"],
            x["whois_created"]
            if x["whois_created"] and x["ctlog_earliest"]
            else None,
        ),
        axis=1,
    )

# set the data type of the domain_to_cert_delta column to float
df["domain_to_earliest_cert_delta"] =
df["domain_to_earliest_cert_delta"].astype("float64")

# domain to latest cert delta
df["domain_to_latest_cert_delta"] = df.apply(
        lambda x: utils.get_days_delta(
            x["ctlog_latest"],
            x["whois_created"]
            if x["whois_created"] and x["ctlog_latest"]
            else None,
        ),
        axis=1,
    )

# set the data type of the domain_to_cert_delta column to float
df["domain_to_latest_cert_delta"] =
df["domain_to_latest_cert_delta"].astype("float64")

# drop columns we imported that we will never use
df = df.drop(columns=["url", "verification_time", "dateadded",
"soa_timestamp","domain_to_cert_delta"])

click.echo(df.head())
click.echo(df.describe(include='all'))
click.echo(df.dtypes)
                            domain  malicious       whois_created
0  i-db5p-cor001.api.p001.1drv.com      False 2013-08-05 18:33:50  \
4        soundcloud-pax.pandora.com      False 1993-12-28 05:00:00
5          joolcomercializadora.com       True 2023-05-22 14:53:50
6         createpdf-asr.acrobat.com      False 1999-03-16 05:00:00
8                           popt.in      False 2016-05-14 16:58:55

       ctlog_earliest        ctlog_latest  ctlog_wildcard
0 2022-01-24 20:01:58 2023-06-08 20:46:06            True  \
4 2022-05-19 00:00:00 2023-06-19 23:59:59            True
5 2022-04-06 22:23:24 2023-09-22 23:59:59           False
6 2022-09-09 00:00:00 2023-10-10 23:59:59            True
8 2023-01-07 20:36:15 2023-08-15 04:16:52           False
```

```
   whois_created_dayofweek  ctlog_earliest_dayofweek
ctlog_latest_dayofweek
0                        0                         0
3  \
4                        1                         3
0
5                        0                         2
4
6                        1                         4
1
8                        5                         5
1

   domain_to_earliest_cert_delta  domain_to_latest_cert_delta
0                        -3095.0                      -3595.0
4                       -10369.0                     -10766.0
5                          410.0                       -124.0
6                        -8578.0                      -8975.0
8                        -2430.0                      -2649.0
                   domain malicious                  whois_created
count               21549     21549                          21549  \
unique              21536         2                            NaN
top     www.mediafire.com     False                            NaN
freq                    2     11739                            NaN
mean                  NaN       NaN  2012-10-03 12:56:32.335050496
min                   NaN       NaN            1986-01-09 00:00:00
25%                   NaN       NaN            2003-05-25 13:35:05
50%                   NaN       NaN            2015-05-07 23:56:05
75%                   NaN       NaN            2023-03-20 15:03:16
max                   NaN       NaN            2023-07-03 08:21:24
std                   NaN       NaN                            NaN

                        ctlog_earliest                    ctlog_latest
count                            21549                           21549  \
unique                             NaN                             NaN
top                                NaN                             NaN
freq                               NaN                             NaN
mean    2022-09-26 15:45:50.943570432  2023-08-14 17:49:06.400900352
min               2021-11-30 05:24:28            2023-01-01 18:42:11
25%               2022-06-24 13:47:12            2023-07-02 08:11:07
50%               2022-10-18 21:00:14            2023-08-21 21:40:11
75%               2022-12-14 00:00:00            2023-09-21 19:41:38
max               2023-06-28 04:36:22            2023-12-31 23:59:59
std                                NaN                             NaN

       ctlog_wildcard  whois_created_dayofweek  ctlog_earliest_dayofweek
count           21549             21549.000000              21549.000000  \
unique              2                      NaN                       NaN
```

```
top              False                   NaN                        NaN
freq             13032                   NaN                        NaN
mean               NaN              2.332823                   2.399462
min                NaN              0.000000                   0.000000
25%                NaN              1.000000                   1.000000
50%                NaN              2.000000                   2.000000
75%                NaN              4.000000                   4.000000
max                NaN              6.000000                   6.000000
std                NaN              1.775043                   1.897252

        ctlog_latest_dayofweek  domain_to_earliest_cert_delta
count             21549.000000                   21549.000000  \
unique                     NaN                            NaN
top                        NaN                            NaN
freq                       NaN                            NaN
mean                  2.873080                   -3645.602070
min                   0.000000                  -13445.000000
25%                   1.000000                   -7078.000000
50%                   3.000000                   -2637.000000
75%                   5.000000                      69.000000
max                   6.000000                     524.000000
std                   2.057394                    3790.677119

        domain_to_latest_cert_delta
count                  21549.000000
unique                          NaN
top                             NaN
freq                            NaN
mean                   -3967.678222
min                   -13798.000000
25%                    -7421.000000
50%                    -3009.000000
75%                     -144.000000
max                      135.000000
std                     3852.703681
domain                                 string[python]
malicious                                        bool
whois_created                          datetime64[ns]
ctlog_earliest                         datetime64[ns]
ctlog_latest                           datetime64[ns]
ctlog_wildcard                                   bool
whois_created_dayofweek                         int64
ctlog_earliest_dayofweek                        int64
ctlog_latest_dayofweek                          int64
domain_to_earliest_cert_delta                 float64
domain_to_latest_cert_delta                   float64
dtype: object
```

In [3]:

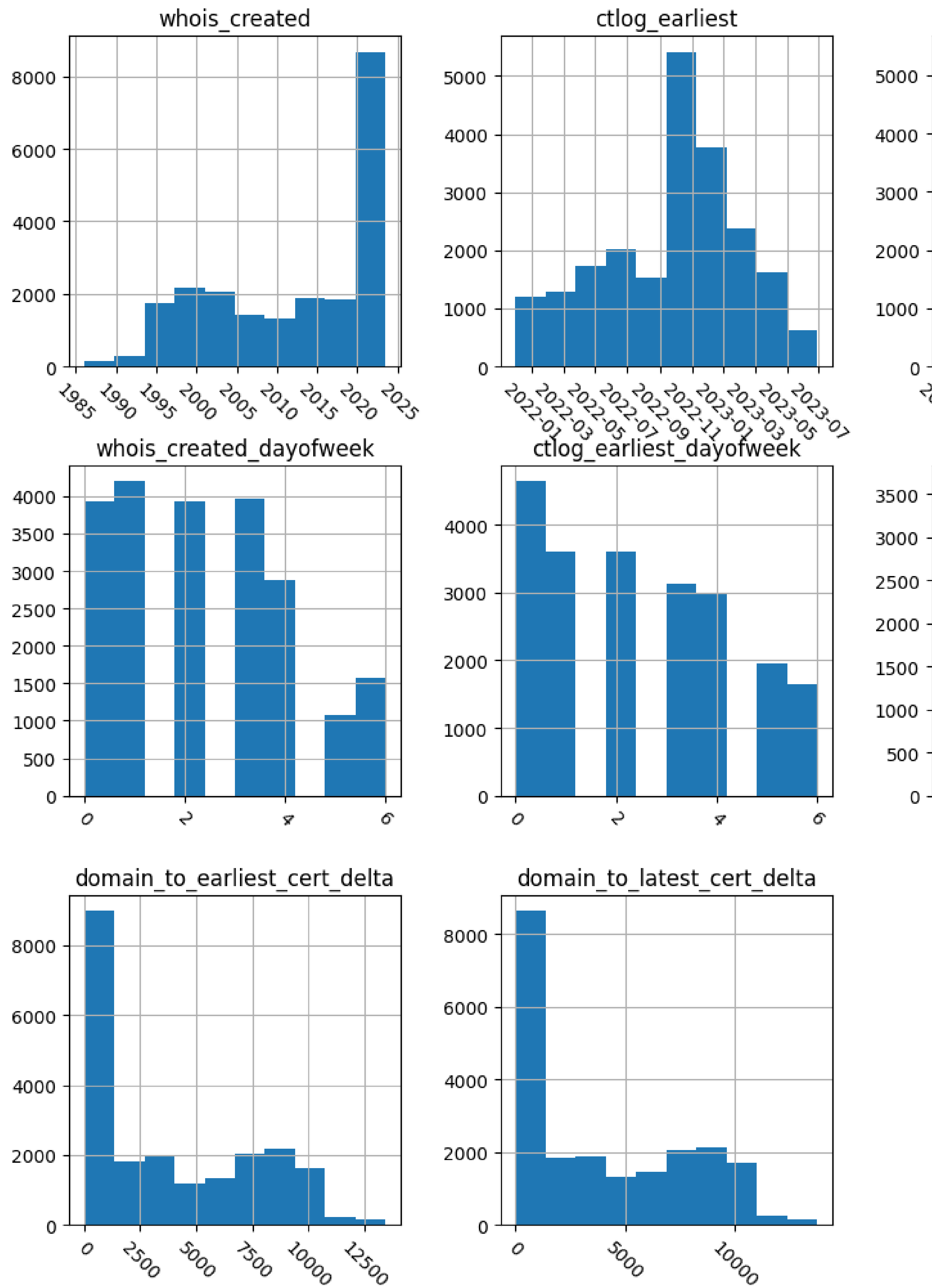
```
# absolute value of the domain_to_cert_delta
```

```
df["domain_to_earliest_cert_delta"] =
abs(df["domain_to_earliest_cert_delta"])
df["domain_to_latest_cert_delta"] = abs(df["domain_to_latest_cert_delta"])

df.hist(figsize=(12,12), xrot=-45)

col_index = df.columns.get_loc("whois_created_dayofweek")
df["whois_created_dow_sin"] = df.apply(lambda row:
math.sin(360/7*(row[col_index])),axis=1)
df["whois_created_dow_cos"] = df.apply(lambda row:
math.cos(360/7*(row[col_index])),axis=1)

col_index = df.columns.get_loc("ctlog_earliest_dayofweek")
df["ctlog_earliest_dow_sin"] = df.apply(lambda row:
math.sin(360/7*(row[col_index])),axis=1)
df["ctlog_earliest_dow_cos"] = df.apply(lambda row:
math.cos(360/7*(row[col_index])),axis=1)

col_index = df.columns.get_loc("ctlog_latest_dayofweek")
df["ctlog_latest_dow_sin"] = df.apply(lambda row:
math.sin(360/7*(row[col_index])),axis=1)
df["ctlog_latest_dow_cos"] = df.apply(lambda row:
math.cos(360/7*(row[col_index])),axis=1)


df.plot.scatter(x="ctlog_earliest_dow_sin",
y="ctlog_earliest_dow_cos").set_aspect('equal')
df.plot.scatter(x="whois_created_dow_sin",
y="whois_created_dow_cos").set_aspect('equal')
df.plot.scatter(x="ctlog_latest_dow_sin",
y="ctlog_latest_dow_cos").set_aspect('equal')

"""
# add one hot encode ctlog_earliest_not_before_dayofweek
df = pd.get_dummies(
        df,
        columns=["ctlog_earliest_dayofweek"],
        prefix=["ctlog_earliest_dow"],
    )
# add one hot encode whois_created_dayofweek to columns
df = pd.get_dummies(
        df, columns=["whois_created_dayofweek"], prefix="whois_dow"
    )
    """

# Summary statistics
click.echo(df.describe(include='all'))
                   domain malicious                  whois_created
count               21549     21549                          21549  \
```

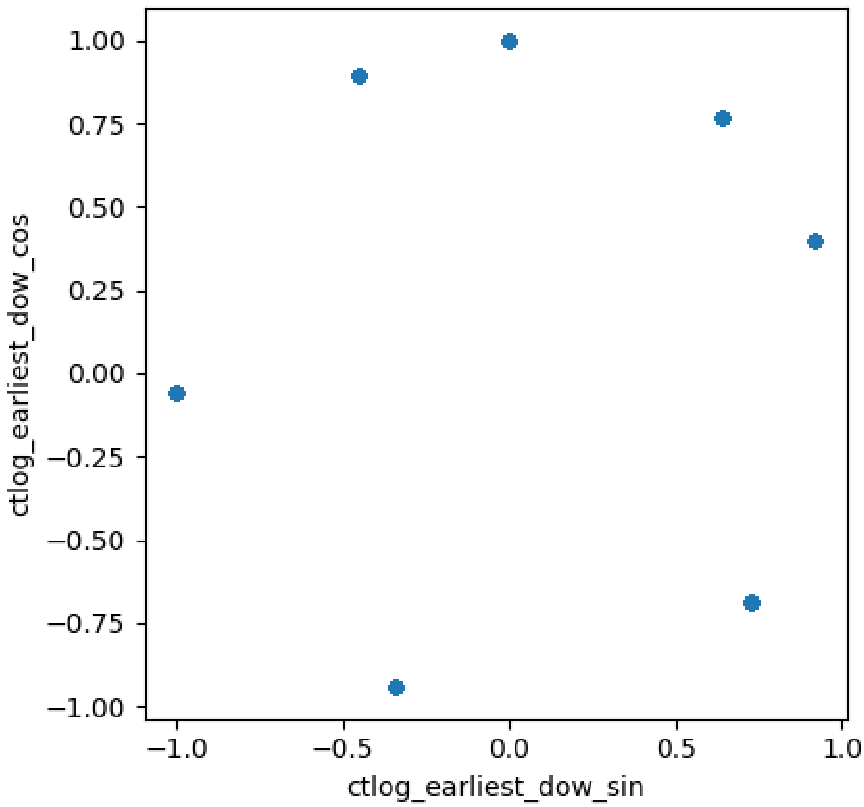
```
unique            21536          2                            NaN
top     www.mediafire.com      False                            NaN
freq                  2      11739                            NaN
mean                NaN        NaN  2012-10-03 12:56:32.335050496
min                 NaN        NaN            1986-01-09 00:00:00
25%                 NaN        NaN            2003-05-25 13:35:05
50%                 NaN        NaN            2015-05-07 23:56:05
75%                 NaN        NaN            2023-03-20 15:03:16
max                 NaN        NaN            2023-07-03 08:21:24
std                 NaN        NaN                            NaN

                        ctlog_earliest                   ctlog_latest
count                            21549                          21549  \
unique                             NaN                            NaN
top                                NaN                            NaN
freq                               NaN                            NaN
mean    2022-09-26 15:45:50.943570432  2023-08-14 17:49:06.400900352
min               2021-11-30 05:24:28            2023-01-01 18:42:11
25%               2022-06-24 13:47:12            2023-07-02 08:11:07
50%               2022-10-18 21:00:14            2023-08-21 21:40:11
75%               2022-12-14 00:00:00            2023-09-21 19:41:38
max               2023-06-28 04:36:22            2023-12-31 23:59:59
std                                NaN                            NaN

       ctlog_wildcard  whois_created_dayofweek  ctlog_earliest_dayofweek
count           21549             21549.000000              21549.000000  \
unique              2                      NaN                       NaN
top             False                      NaN                       NaN
freq            13032                      NaN                       NaN
mean              NaN                 2.332823                  2.399462
min               NaN                 0.000000                  0.000000
25%               NaN                 1.000000                  1.000000
50%               NaN                 2.000000                  2.000000
75%               NaN                 4.000000                  4.000000
max               NaN                 6.000000                  6.000000
std               NaN                 1.775043                  1.897252

        ctlog_latest_dayofweek  domain_to_earliest_cert_delta
count             21549.000000                   21549.000000  \
unique                     NaN                            NaN
top                        NaN                            NaN
freq                       NaN                            NaN
mean                  2.873080                    3742.948397
min                   0.000000                       0.000000
25%                   1.000000                     181.000000
50%                   3.000000                    2637.000000
75%                   5.000000                    7078.000000
max                   6.000000                   13445.000000
std                   2.057394                    3694.584062
```

```
        domain_to_latest_cert_delta  whois_created_dow_sin
count                  21549.000000           21549.000000  \
unique                          NaN                    NaN
top                             NaN                    NaN
freq                            NaN                    NaN
mean                    3969.491206               0.140419
min                        0.000000              -0.998199
25%                      144.000000              -0.340712
50%                     3009.000000               0.000000
75%                     7421.000000               0.728010
max                    13798.000000               0.918032
std                     3850.835626               0.659922

        whois_created_dow_cos  ctlog_earliest_dow_sin
ctlog_earliest_dow_cos
count            21549.000000            21549.000000
21549.000000  \
unique                    NaN                     NaN
NaN
top                       NaN                     NaN
NaN
freq                      NaN                     NaN
NaN
mean                 0.054288                0.095357
0.161451
min                 -0.940168               -0.998199               -
0.940168
25%                 -0.685567               -0.340712               -
0.685567
50%                  0.396506                0.000000
0.396506
75%                  0.767830                0.728010
0.892589
max                  1.000000                0.918032
1.000000
std                  0.736128                0.651782
0.734891

        ctlog_latest_dow_sin  ctlog_latest_dow_cos
count           21549.000000          21549.000000
unique                   NaN                   NaN
top                      NaN                   NaN
freq                     NaN                   NaN
mean                0.096253              0.255578
min                -0.998199             -0.940168
25%                -0.450871             -0.685567
50%                 0.000000              0.396506
75%                 0.728010              0.892589
```

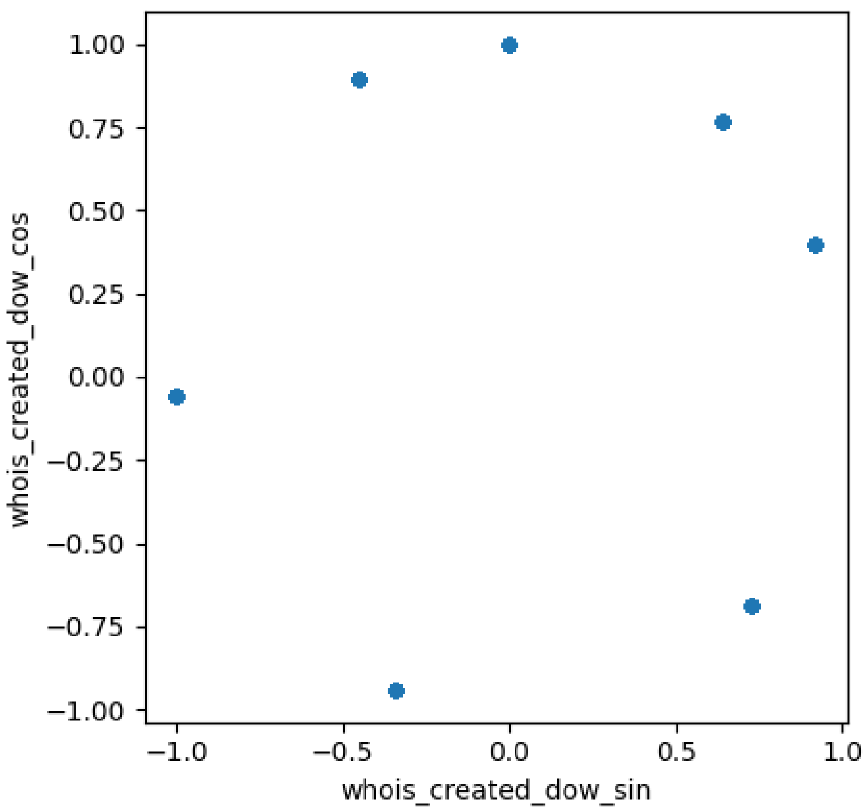
```
max                 0.918032            1.000000
std                 0.651597            0.707728
```



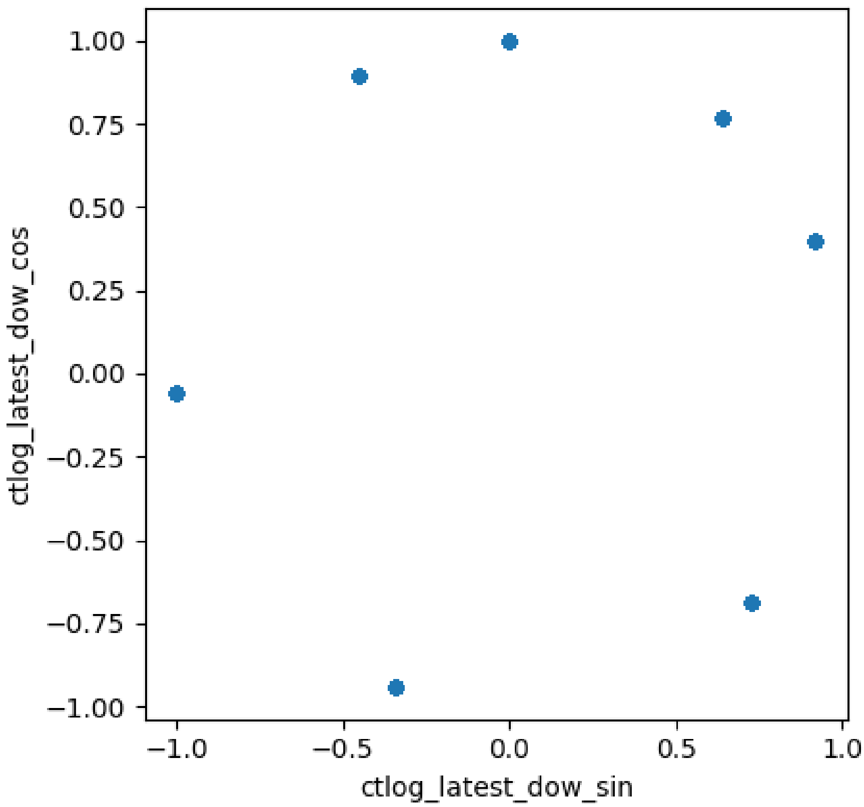
whois_created
ctlog_earliest
whois_created_dayofweek
ctlog_earliest_dayofweek
domain_to_earliest_cert_delta
domain_to_latest_cert_delta



1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
ctlog_earliest_dow_cos
−1.0
−0.5
0.0
0.5
1.0
ctlog_earliest_dow_sin



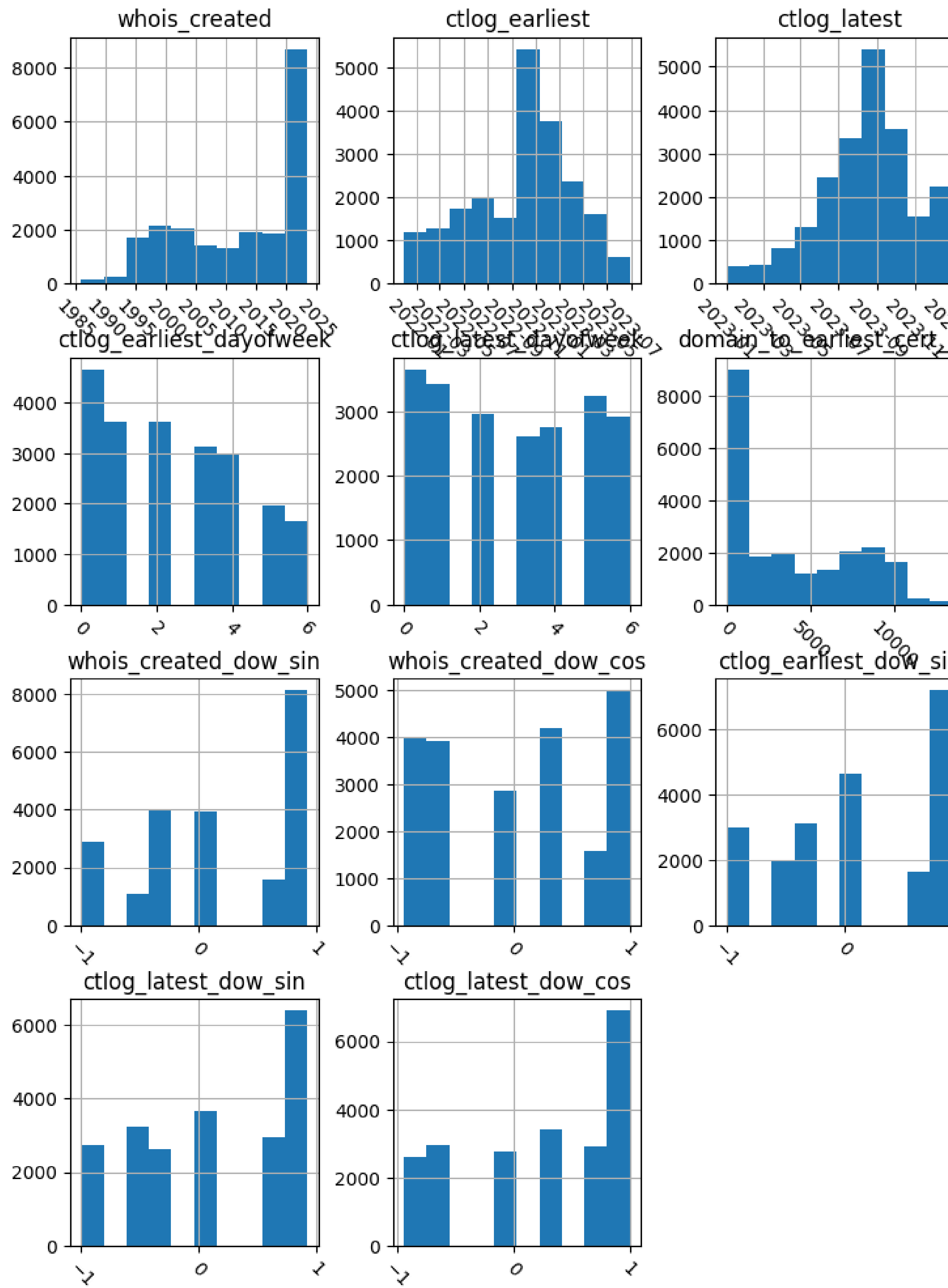
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
whois_created_dow_cos
−1.0
−0.5
0.0
0.5
1.0
whois_created_dow_sin



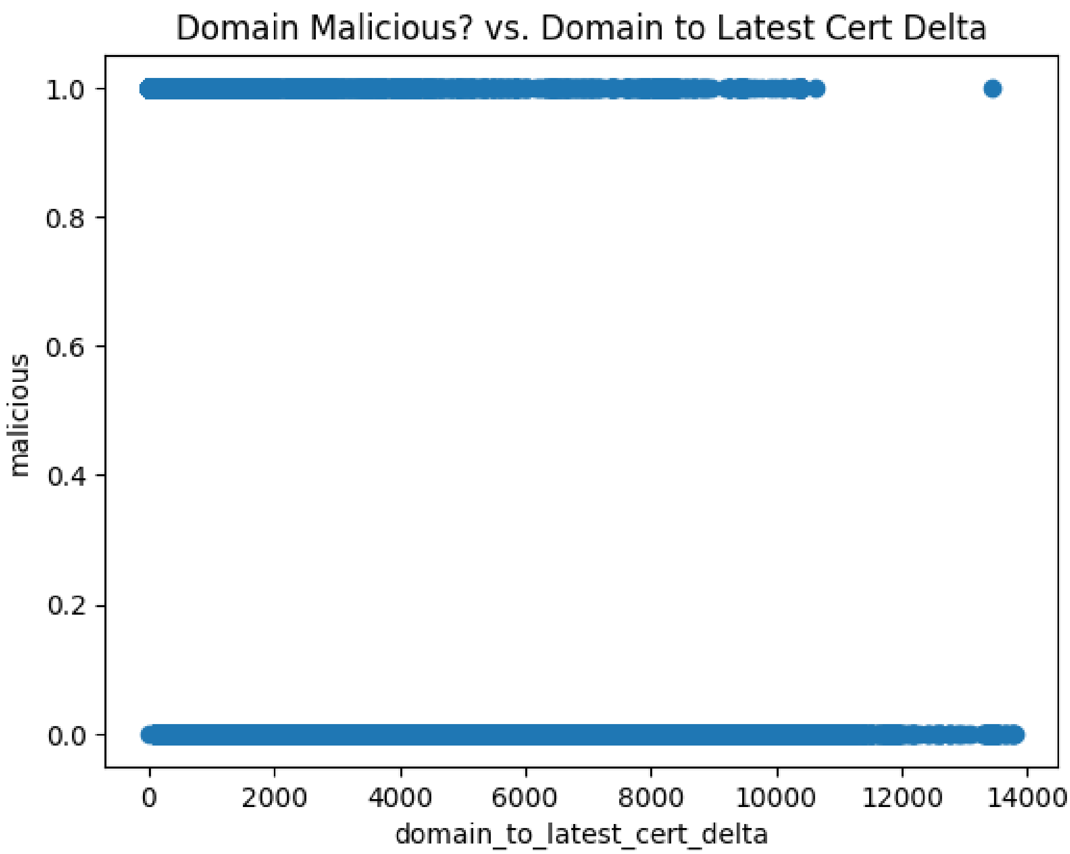
In [4]:

```
# Plot histograms
df.hist(figsize=(12,12), xrot=-45)

# Create a scatter plot of the data, showing malicious and benign domains
# plot the data as a scatter plot
plt.scatter(df["domain_to_earliest_cert_delta"], df["malicious"])
plt.xlabel("domain_to_earliest_cert_delta")
plt.ylabel("malicious")
plt.title("Domain Malicious? vs. Domain to Earliest Cert Delta")
plt.show()
# and for the latest
plt.scatter(df["domain_to_latest_cert_delta"], df["malicious"])
plt.xlabel("domain_to_latest_cert_delta")
plt.ylabel("malicious")
plt.title("Domain Malicious? vs. Domain to Latest Cert Delta")
plt.show()



click.echo(df.head())

# print a count of malicious and benign values
click.echo(df["malicious"].value_counts())
# deduplicate any rows, there shouldn't be any, but just in case
df = df.drop_duplicates()
click.echo(df["malicious"].value_counts())
```

```
click.echo(df.head())

X = df.drop(["malicious","domain", "ctlog_earliest", "ctlog_latest",
"whois_created", "whois_created_dayofweek", "ctlog_earliest_dayofweek"],
axis=1)
X = df.filter(combo_features)
y = df.filter(["malicious"])


click.echo(X.describe())
```



whois_created
ctlog_earliest
ctlog_latest
ctlog_earliest_dayofweek
ctlog_latest_dayofweek
domain_to_earliest_cert_
whois_created_dow_sin
whois_created_dow_cos
ctlog_earliest_dow_si
ctlog_latest_dow_sin
ctlog_latest_dow_cos
0
2000
4000
6000
8000
1000
3000
5000
1985
1990
1995
2000
2005
2010
2015
2020
2025
2
4
6
10000
-1
1




```
                          domain  malicious       whois_created
0  i-db5p-cor001.api.p001.1drv.com      False 2013-08-05 18:33:50  \
4       soundcloud-pax.pandora.com      False 1993-12-28 05:00:00
5         joolcomercializadora.com       True 2023-05-22 14:53:50
6        createpdf-asr.acrobat.com      False 1999-03-16 05:00:00
8                          popt.in      False 2016-05-14 16:58:55

       ctlog_earliest         ctlog_latest  ctlog_wildcard
0 2022-01-24 20:01:58 2023-06-08 20:46:06            True  \
4 2022-05-19 00:00:00 2023-06-19 23:59:59            True
5 2022-04-06 22:23:24 2023-09-22 23:59:59           False
6 2022-09-09 00:00:00 2023-10-10 23:59:59            True
8 2023-01-07 20:36:15 2023-08-15 04:16:52           False

   whois_created_dayofweek  ctlog_earliest_dayofweek
ctlog_latest_dayofweek
0                        0                          0
3  \
4                        1                          3
0
5                        0                          2
4
6                        1                          4
1
8                        5                          5
1
```

```
   domain_to_earliest_cert_delta  domain_to_latest_cert_delta
0                         3095.0                       3595.0  \
4                        10369.0                      10766.0
5                          410.0                        124.0
6                         8578.0                       8975.0
8                         2430.0                       2649.0

   whois_created_dow_sin  whois_created_dow_cos  ctlog_earliest_dow_sin
0               0.000000               1.000000                0.000000  \
4               0.918032               0.396506               -0.340712
5               0.000000               1.000000                0.728010
6               0.918032               0.396506               -0.998199
8              -0.450871               0.892589               -0.450871

   ctlog_earliest_dow_cos  ctlog_latest_dow_sin  ctlog_latest_dow_cos
0                1.000000             -0.340712             -0.940168
4               -0.940168              0.000000              1.000000
5               -0.685567             -0.998199             -0.059997
6               -0.059997              0.918032              0.396506
8                0.892589              0.918032              0.396506
malicious
False    11739
True      9810
Name: count, dtype: int64
malicious
False    11739
True      9810
Name: count, dtype: int64
                            domain  malicious       whois_created
0  i-db5p-cor001.api.p001.1drv.com      False 2013-08-05 18:33:50  \
4       soundcloud-pax.pandora.com      False 1993-12-28 05:00:00
5         joolcomercializadora.com       True 2023-05-22 14:53:50
6        createpdf-asr.acrobat.com      False 1999-03-16 05:00:00
8                          popt.in      False 2016-05-14 16:58:55

       ctlog_earliest        ctlog_latest  ctlog_wildcard
0 2022-01-24 20:01:58 2023-06-08 20:46:06            True  \
4 2022-05-19 00:00:00 2023-06-19 23:59:59            True
5 2022-04-06 22:23:24 2023-09-22 23:59:59           False
6 2022-09-09 00:00:00 2023-10-10 23:59:59            True
8 2023-01-07 20:36:15 2023-08-15 04:16:52           False

   whois_created_dayofweek  ctlog_earliest_dayofweek
ctlog_latest_dayofweek
0                        0                         0
3  \
4                        1                         3
0
```

```
5                        0                         2
4
6                        1                         4
1
8                        5                         5
1

   domain_to_earliest_cert_delta  domain_to_latest_cert_delta
0                         3095.0                       3595.0  \
4                        10369.0                      10766.0
5                          410.0                        124.0
6                         8578.0                       8975.0
8                         2430.0                       2649.0

   whois_created_dow_sin  whois_created_dow_cos  ctlog_earliest_dow_sin
0               0.000000               1.000000                0.000000  \
4               0.918032               0.396506               -0.340712
5               0.000000               1.000000                0.728010
6               0.918032               0.396506               -0.998199
8              -0.450871               0.892589               -0.450871

   ctlog_earliest_dow_cos  ctlog_latest_dow_sin  ctlog_latest_dow_cos
0                1.000000             -0.340712             -0.940168
4               -0.940168              0.000000              1.000000
5               -0.685567             -0.998199             -0.059997
6               -0.059997              0.918032              0.396506
8                0.892589              0.918032              0.396506
       domain_to_earliest_cert_delta  ctlog_earliest_dow_sin
count                   21549.000000            21549.000000  \
mean                     3742.948397                0.095357
std                      3694.584062                0.651782
min                         0.000000               -0.998199
25%                       181.000000               -0.340712
50%                      2637.000000                0.000000
75%                      7078.000000                0.728010
max                     13445.000000                0.918032

       ctlog_earliest_dow_cos  whois_created_dow_sin  whois_created_dow_cos
count            21549.000000           21549.000000           21549.000000
\
mean                 0.161451               0.140419               0.054288
std                  0.734891               0.659922               0.736128
min                 -0.940168              -0.998199              -0.940168
25%                 -0.685567              -0.340712              -0.685567
50%                  0.396506               0.000000               0.396506
75%                  0.892589               0.728010               0.767830
max                  1.000000               0.918032               1.000000
```

```
       domain_to_latest_cert_delta  ctlog_latest_dow_sin
ctlog_latest_dow_cos
count                   21549.000000          21549.000000
21549.000000
mean                     3969.491206              0.096253
0.255578
std                      3850.835626              0.651597
0.707728
min                         0.000000             -0.998199             -
0.940168
25%                       144.000000             -0.450871             -
0.685567
50%                      3009.000000              0.000000
0.396506
75%                      7421.000000              0.728010
0.892589
max                     13798.000000              0.918032
1.000000
```

In [5]:

```
# convert y (malicious) to 1/0 int
y = y.astype('int')
# convert X["ctlog_wildcard"]  to 1/0 int
if "ctlog_wildcard" in X.columns:
    X["ctlog_wildcard"] = X["ctlog_wildcard"].astype('int')

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)


# random forest model

param_grid = {
    'n_estimators': [50,100,150,200],
    'max_features': ['sqrt', 'log2'],
    'max_depth' : [2,3,4,5],
    'criterion' :['gini', 'entropy']
}
```

In [6]:

```
rf = RandomForestClassifier(random_state=42)
rf_cv = GridSearchCV(estimator=rf, param_grid=param_grid, cv=5, n_jobs=-1)
rf_cv.fit(X_train, y_train.values.ravel())
```

Out[6]:

GridSearchCV

```
GridSearchCV(cv=5, estimator=RandomForestClassifier(random_state=42),
n_jobs=-1,
             param_grid={'criterion': ['gini', 'entropy'],
                         'max_depth': [2, 3, 4, 5],
                         'max_features': ['sqrt', 'log2'],
                         'n_estimators': [50, 100, 150, 200]})
```

estimator: RandomForestClassifier

```
RandomForestClassifier(random_state=42)
```

```
RandomForestClassifier
RandomForestClassifier(random_state=42)
```

In [7]:

```
bp = rf_cv.best_params_
click.echo("Best parameters set found:")
click.echo(bp)
```

```
Best parameters set found:
{'criterion': 'gini', 'max_depth': 5, 'max_features': 'sqrt',
'n_estimators': 50}
```

In [8]:

```
rf = RandomForestClassifier(random_state=42,
max_features=bp["max_features"], n_estimators=bp["n_estimators"],
max_depth=bp["max_depth"], criterion=bp["criterion"])
```

In [9]:

```
rf.fit(X_train, y_train.values.ravel())
```

Out[9]:

```
RandomForestClassifier
RandomForestClassifier(max_depth=5, n_estimators=50, random_state=42)
```

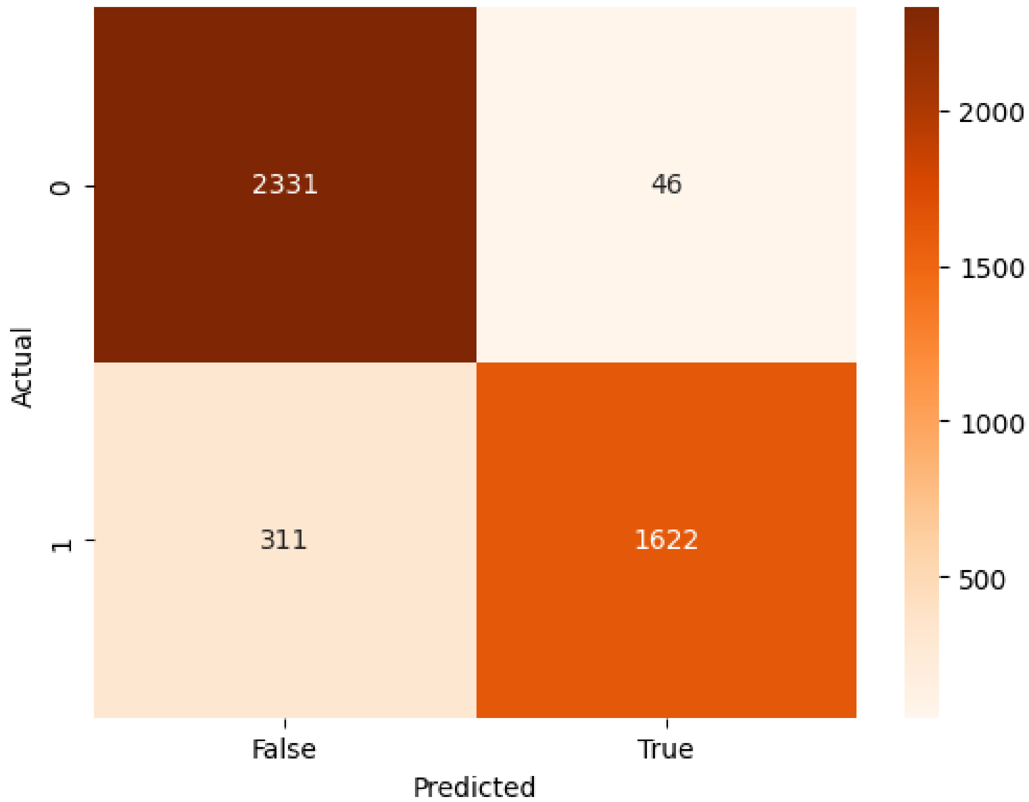
In [10]:

```
# Predict the malicious column using the test data
#add the incepts


y_predicted = rf.predict(X_test)

# Present the results
click.echo("Features selected:")
click.echo(X.columns)
click.echo("Confusion matrix:")
cm = confusion_matrix(y_test, y_predicted)
click.echo(cm)
click.echo("Classification report:")
click.echo(classification_report(y_test, y_predicted))


# Heatmap of confusion matrix
y_predicted

threshold = 0.5
y_predicted = [y > threshold for y in y_predicted]
data = {'Actual':    y_test.values.flatten(),
        'Predicted': y_predicted
        }

# Generate a confusion matrix and heatmap to evaluate the Type I and Type
II errors/ FP/FN etc.
df = pd.DataFrame(data, columns=['Actual','Predicted'])
```

```
cm2 = pd.crosstab(df['Actual'], df['Predicted'], rownames=['Actual'],
colnames=['Predicted'])
fig = sns.heatmap(cm2, annot=True, cmap='Oranges', fmt='g')
fig
Features selected:
Index(['domain_to_earliest_cert_delta', 'ctlog_earliest_dow_sin',
       'ctlog_earliest_dow_cos', 'ctlog_wildcard', 'whois_created_dow_sin',
       'whois_created_dow_cos', 'domain_to_latest_cert_delta',
       'ctlog_latest_dow_sin', 'ctlog_latest_dow_cos'],
      dtype='object')
Confusion matrix:
[[2331   46]
 [ 311 1622]]
Classification report:
              precision    recall  f1-score   support

           0       0.88      0.98      0.93      2377
           1       0.97      0.84      0.90      1933

    accuracy                           0.92      4310
   macro avg       0.93      0.91      0.91      4310
weighted avg       0.92      0.92      0.92      4310
```

Out[10]:

```
<Axes: xlabel='Predicted', ylabel='Actual'>
```



In [11]:

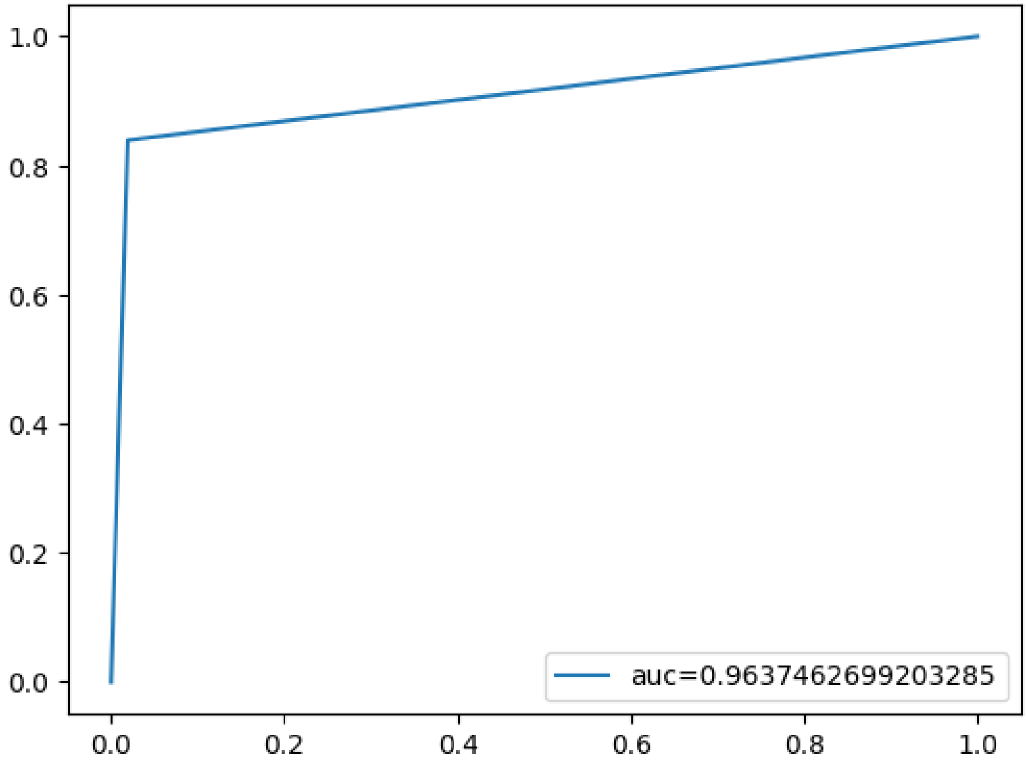
```
rf_cv.score(X_test, y_test.values.ravel())
```

Out[11]:

```
0.917169373549884
```

In [12]:

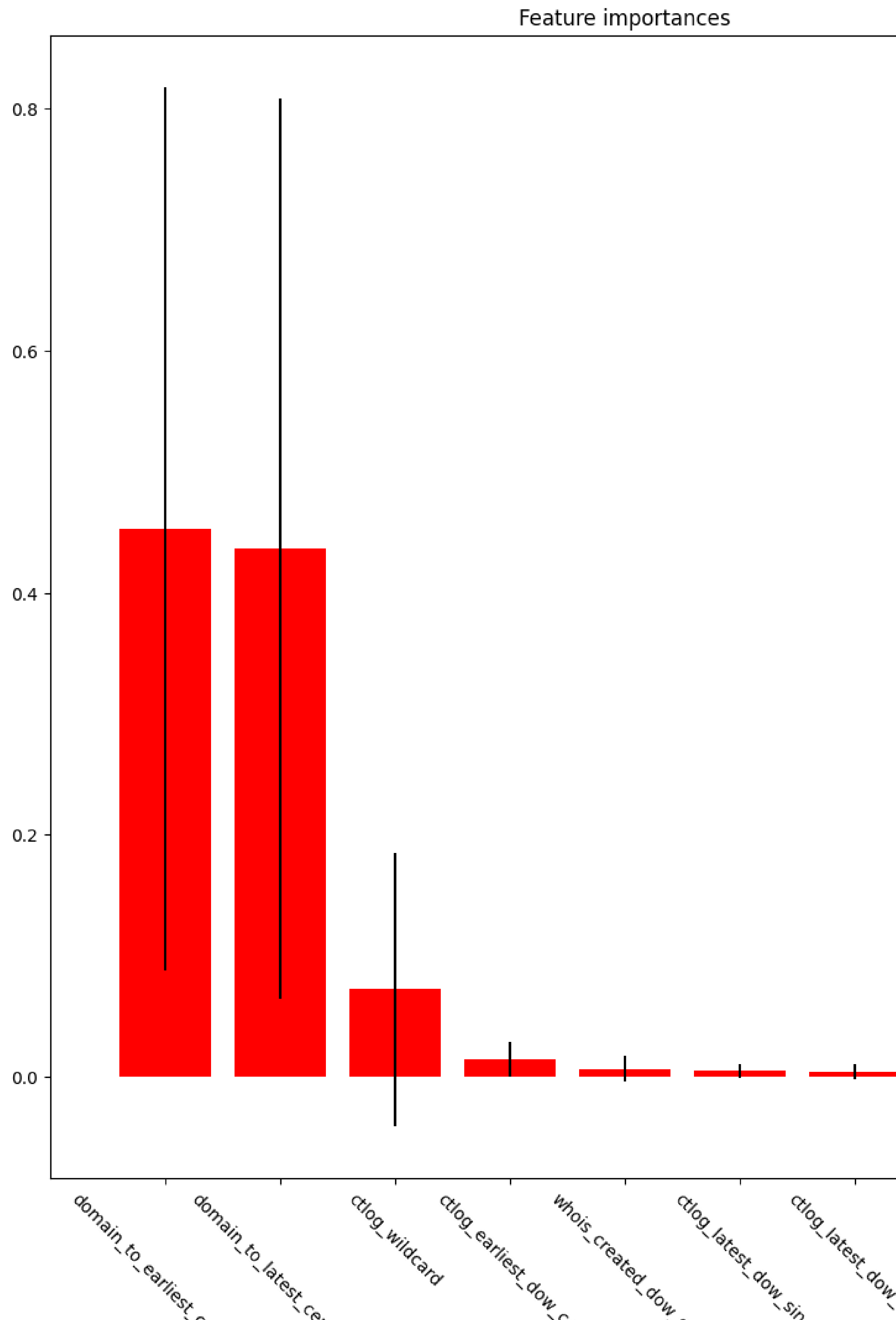
```
# Receiver Operating Characteristic (ROC) curve [True Pos vs False Pos],
measure the area under the curve.
y_pred_proba = rf.predict_proba(X_test)[::,1]
fpr, tpr, thresholds = roc_curve(y_test, y_predicted)
auc = roc_auc_score(y_test, y_pred_proba)
plt.plot(fpr,tpr,label="auc="+str(auc))
plt.legend(loc=4)
plt.show()
```



In [13]:

```
# plot the feature importances
importances = rf.feature_importances_
std = np.std([tree.feature_importances_ for tree in rf.estimators_],
axis=0)

indices = np.argsort(importances)[::-1]
print(indices)
# Print the feature ranking
click.echo("Feature ranking:")
click.echo(X.shape[1])
for f in range(X.shape[1]):
    # get the feature name from the combo_features list
    click.echo("%d. feature %s (%f)" % (f + 1, combo_features[indices[f]],
importances[indices[f]]))

# Plot the feature importances of the forest
```

```
plt.figure(figsize=(12,12))
plt.title("Feature importances")
plt.bar(range(X.shape[1]), importances[indices], color="r",
yerr=std[indices], align="center")
plt.xticks(range(X.shape[1]), X.columns[indices], rotation=-45)
plt.xlim([-1, X.shape[1]])
plt.show()
[0 6 3 2 4 7 8 1 5]
Feature ranking:
9
1. feature domain_to_earliest_cert_delta (0.452833)
2. feature domain_to_latest_cert_delta (0.436609)
3. feature ctlog_wildcard (0.072233)
4. feature ctlog_earliest_dow_cos (0.014518)
5. feature whois_created_dow_sin (0.006729)
6. feature ctlog_latest_dow_sin (0.004943)
7. feature ctlog_latest_dow_cos (0.004507)
8. feature ctlog_earliest_dow_sin (0.003904)
9. feature whois_created_dow_cos (0.003724)
```



Feature importances
0.8
0.6
0.4
0.2
0.0
ctlog_wildcard

In [14]:

```
from sklearn.inspection import permutation_importance

result = permutation_importance(rf, X_test, y_test, n_repeats=100,
random_state=42, n_jobs=-1)

forest_importances = pd.Series(result.importances_mean, index=X.columns)
fig, ax = plt.subplots()
forest_importances.plot.bar(yerr=result.importances_std, ax=ax)
ax.set_title("Feature importances using permutation")
ax.set_ylabel("Mean accuracy decrease")
fig.tight_layout()
plt.show()
```