Configuration Manual

MSc Research Project
Cyber Security

Keerthi Prabhakar
Student ID: x19211023

School of Computing
National College of Ireland

Supervisor: Imran Khan

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	KEERTHI PRABHAKAR		
Student ID:	x19211023		
Programme:	MSc in Cybersecurity	**Year:**	2020-2021
Module:	MSc Internship		
Supervisor:	Imran Khan		
Submission Due Date:	16/12/2021		
Project Title:	A Proactive Approach to Predict Phishing Websites		
Word Count:	1100	**Page Count**:	7

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.
<u>ALL</u> 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: ………

Date: …………16/12/2021…………

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):	

Contents

List of Figures

Configuration Manual

Keerthi Prabhakar
X19211023

1. Introduction

This research mainly aims to predict phishing websites using Machine Learning and Deep learning algorithms. The project has 3 phases, first data collection, second feature extraction and the machine and deep learning models are the third phase. This manual will provide the required instructions to execute this project to get the desired result.

2. Requirements

2.1 System Requirements

- Laptop – Windows/Mac/Linux Machine
- RAM: 8gb DDR4
- MS Excel – for analysing the datasets.
- Web Browser – Chrome for best results.
- Internet Connection

Note: The experiment was conducted on a windows machine and latest version of Google Chrome browser (Version 96.0.4664.45).

2.2 Software requirements

- Google Colab – Recommended (*Google Colaboratory*, 2021)
- Anaconda – 64 bit.
- Python 3

Note: This experiment has been conducted on Google Colab Notebook, if the experiment needs to be performed on a local system it is recommended to use Anaconda or Python 3

2.3 Packages and Imports

Models were deployed using google colab Notebook, the programming language used is Python. Below are the packages and imports that are required for the models to function:

- pandas
- numpy
- sklearn
- requests
- urllib
- ipaddress
- BeautifulSoup
- whois
- datetime
- matplotlib
- seaborn
- tensorflow
- Fast

3. Dataset Collection

For this project, we need a bunch of URL's of type genuine and malicious.

The collection of phishing urls is downloaded from Phishtank(*PhishTank* , 2021) an open source phishing links repository.

For the legitimate URLs, the source of the dataset is University of New Brunswick(*University of New Brunswick*, 2021).

3.1 Feature Extraction

In this phase, python code is developed to extract relevant 30 features to analyse the authenticity of the URL. This is the initial step which is also termed as Data Pre-processing. This program yields a csv file that contains numerical values which gives insights about the URL. The output is then fed to the ML and DL models.

Here are few examples of feature extraction:

a) Domain Age

```
def domainAge(domain_name):
  creation_date = domain_name.creation_date
  expiration_date = domain_name.expiration_date
  if (isinstance(creation_date,str) or isinstance(expiration_date,str)):
    try:
      creation_date = datetime.strptime(creation_date,'%Y-%m-%d')
      expiration_date = datetime.strptime(expiration_date,"%Y-%m-%d")
    except:
      return 1
  if ((expiration_date is None) or (creation_date is None)):
      return 1
  elif ((type(expiration_date) is list) or (type(creation_date) is list)):
      return 1
  else:
    ageofdomain = abs((expiration_date - creation_date).days)
    if ((ageofdomain/30) < 6):
      age = 1
    else:
      age = 0
  return age
```

Figure 1: Domain Age Feature Extraction

b) Evaluation of " @" symbol in the URL:

```python
def haveAtSign(url):
  if "@" in url:
    at = 1
  else:
    at = 0
  return at
```

Figure 2: @ feature extraction

c) iFrame

```python
def iframe(response):
  if response == "":
      return 0
  else:
      if re.findall(r"[<iframe>|<frameBorder>]", response.text):
          return 1
      else:
          return 0
```

Figure 3: Iframe Feature Extraction

d) Right Click

```python
def RightClick(response):
  if response == "":
    return 1
  else:
    if re.findall(r"event.button ?== ?2", response.text):
      return 0
    else:
      return 1
```

Figure 4: Right-Click Feature Extraction

Likewise total ***30 features*** are extracted and a .csv file will be generated which is then fed to deep and machine learning models.

4. Implementation

Insights into Data

Upload the .csv file that was generated by the python code in feature extraction to the ML and DL models.

```
df = pd.read_csv("dataset(50-50).csv")
```

	Shortining_Service	popUpWindow	SSLfinal_State	URL_Length	RightClick	port	HTTPS_token	Prefix_Suffix	SFH	Links_in_tags	Iframe	DNSRecord	Google_Index	doubl
0	0	0	0	1	1	0	0	0	1	0	1	1	1	
1	0	0	0	1	1	0	0	0	1	1	1	1	0	
2	0	1	0	1	1	0	0	0	1	1	1	1	1	
3	0	0	0	1	1	0	0	0	1	0	1	1	0	
4	0	0	0	1	1	0	0	0	1	1	1	1	1	

Figure 5: Reading the dataset

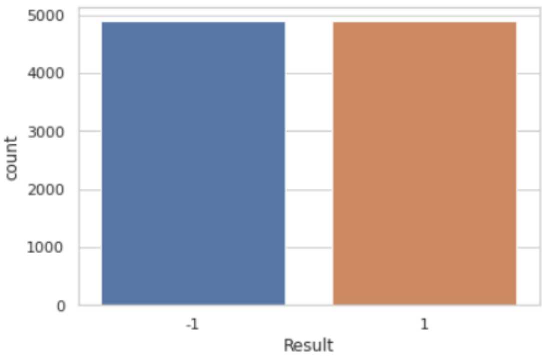

Figure 6:Visualizing the data

Data Correlation :

The below function gives the correlation matrix of the dataset

```
def correlation(dataset, threshold):
    col_corr = set()  # Set of all the names of correlated columns
    corr_matrix = df.corr()
    for i in range(len(corr_matrix.columns)):
        for j in range(i):
            if abs(corr_matrix.iloc[i, j]) > threshold: # absolute coeff value
                colname = corr_matrix.columns[i]  # extracting the name of column
                col_corr.add(colname)
    return col_corr
```

Figure 7:Data correlation

4.1 Machine Learning Model

An ensemble model is developed using Deep learning and Machine Learning approaches. The dataset is fed to machine learning and deep learning models and the accuracy of the algorithms provides the capacity of the algorithm to predict malicious url's.

1) Logistic regression

Using SKlearn model selection library we can automatically import LogisticRegression Function.

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.20,random_state=0)
```

```
[ ] from sklearn.linear_model import LogisticRegression
```

```
[ ] logreg = LogisticRegression()
```

```
[ ] logreg.fit(X_train,y_train)

LogisticRegression()
```

```
y_pred=logreg.predict(X_test)
```

```
[ ]
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import metrics
from sklearn.metrics import classification_report, confusion_matrix

# Use score method to get accuracy of model
score = logreg.score(X_test, y_test)
print(score)
```

Figure 8: Logistic Regression

2) Decision Tree Model

Using SKlearn model selection library we can automatically import DecisionTreeClassifier Function.

```
[ ] from sklearn import tree
    clf = tree.DecisionTreeClassifier()
    clf = clf.fit(X_train, y_train)

[ ] y_pred = clf.predict(X_test)
    acc_dec = metrics.accuracy_score(y_test, y_pred)

[ ] print("Accuracy:",metrics.accuracy_score(y_test, y_pred))

    Accuracy: 0.9857084524295631

    cm_tree = metrics.confusion_matrix(y_test, y_pred)
```

Figure 9: Decision Tree

3) Random Forest

Using SKlearn model selection library we can automatically import RandomForestClassifier Function.

```
[ ] from sklearn.ensemble import RandomForestClassifier
    from sklearn.datasets import make_classification

[ ] clf=RandomForestClassifier(n_estimators=100)

[ ] clf.fit(X_train,y_train)

    RandomForestClassifier()

[ ] y_pred=clf.predict(X_test)

    y_pred
```

Figure 10: Random Forest

4.2 Deep Learning Implementation

a) FastAi

Data is normalised and split Tran to test ratio as 80:20 using StratifiedSFold.split Function and then Fed to Fastai model.

```
procs = [FillMissing, Categorify, Normalize]
StratifiedSFold = StratifiedShuffleSplit(n_splits = 1, test_size=0.2, random_state=0)
print(StratifiedSFold)

StratifiedShuffleSplit(n_splits=1, random_state=0, test_size=0.2,
            train_size=None)
```

```
for train, test in StratifiedSFold.split(df.index, df[final_var]):
    data_fold = (TabularList.from_df(df, path=dataloc, cat_names=cat_names, cont_names=cont_names, procs=procs)
                     .split_by_idxs(train, test)
                     .label_from_df(cols=final_var)
                     .databunch())
    # create model and learn
    model = tabular_learner(data_fold, layers=[200, 100], metrics=accuracy, callback_fns=ShowGraph)
    model.fit_one_cycle(cyc_len=10) #
```

Figure 11: Fastai Model

b) CNN using Keras Tensorflow

The CNN model is loaded using keras tensflow library. Non-linear activation function, Rectified Linear Unit is employed as it does not activate all nodes in the model which is beneficial during the backpropagation process. To standarize the inputs BatchNormalization function is used.

```
def TESTbaseline_model(inputDim=-1,outputDim=-1):
    global model_extension, experimentTitle
    model = tf.keras.Sequential([
        Dense(128, activation='relu', input_shape=(inputDim,)),
        BatchNormalization(),
        Dense(64, activation='relu'),
        BatchNormalization(),
        Dense(outputDim, activation='softmax')])
    model_extension = "_binary"
    experimentTitle = "Binary"
    model.compile(optimizer='adam',loss='binary_crossentropy',metrics=['accuracy'])
    return model
```

Figure 12: CNN BatchNormalization and Relu Activation

CNN- Keras Tensflow prediction model

```python
def Dropoutbaseline_model(inputDim=-1,outputDim=-1):
    global model_extension, experimentTitle
    model = tf.keras.Sequential([
        Dense(128, activation='relu', input_shape=(inputDim,)),
        BatchNormalization(),
        Dropout(.5),
        Dense(64, activation='relu'),
        BatchNormalization(),
        Dropout(.5),
        Dense(outputDim, activation='softmax')
    ]) #This is the output layer
    model_extension = "_binary"
    experimentTitle = "Binary"
    model.compile(optimizer='adam',loss='binary_crossentropy',metrics=['accuracy'])
def DPexperiment(dataframe, early=False):
    kfold = StratifiedKFold(n_splits=10, shuffle=True, random_state=seed)
    encoded_y = dataframe.copy()
    encoded_y = encode_labels(encoded_y)
    X=StandardScaler().fit_transform(dataframe.drop(dep_var, axis=1))
    y=LabelEncoder().fit_transform(dataframe[dep_var].values)
    start_time = time.time()
    for index, (train_indices, val_indices) in enumerate(kfold.split(X, y)):
        xtrain, xval = X[train_indices], X[val_indices]
        ytrain, yval = encoded_y[train_indices], encoded_y[val_indices]
        inputDim=xtrain.shape[1]
        outputDim=ytrain.shape[1]
        print("Running fold #" + str(index+1))
        model = TESTbaseline_model(inputDim,outputDim)
        time_gen = int(time.time())
        global model_name
        model_name = f"{dataFile}_{time_gen}"
        tensorboard = TensorBoard(log_dir='keras_tensorflow_logs/{}/{}_{}'.format(experimentTitle, model_name, model_extension),update_freq='epoch')
        if early_stop:
            callbacks = [tensorboard,early_stop]
        else:
            callbacks = [tensorboard, early_stop]
        history = model.fit(xtrain, ytrain, epochs=epochs, validation_data=(xval,yval), callbacks=callbacks, batch_size=batch_size, verbose=0)
    global end_time
    end_time = time.time() - start_time
    TimeSetup = str(datetime.timedelta(seconds=end_time))
    Minutes = int(end_time/60)
    print("Time to complete {} [hour:min:sec]".format(TimeSetup))
    return model, history, X, encoded_y
```

Figure 13: CNN using Keras Tensorflow

5. Execution Steps

1. Download all the .ipynb files from the zip folder uploaded on moodle- Anti_Phishing- URL Feature Extraction.ipynb, Anti_Phishing- Machine Learning.ipynb, Anti_Phishing-Deep Learning.ipynb
2. Open these files individually on Google colab notebook online.
3. Execute the file by using Run all option or run individual cells. This generates a .csv file (dataset.csv).
4. Load this dataset.csv file to Google Colab notebooks (Anti_Phishing- Machine Learning.ipynb, Anti_Phishing-Deep Learning.ipynb)
5. Execute the notebooks by run all option or run individual cells (Anti_Phishing- Machine Learning.ipynb, Anti_Phishing-Deep Learning.ipynb

The video presentation can be accessed here:

https://studentncirl.sharepoint.com/:v:/s/Kee/Ec-cx3n89apDsBPlF3GkaQQB6DT6yjtVlX6XodY20r3nmQ?e=85DNXd

References

Google Colaboratory (2021). Available at: https://colab.research.google.com/?utm_source=scs-index (Accessed: 11 December 2021).

PhishTank (no date). Available at: https://www.phishtank.com/developer_info.php (Accessed: 6 December 2021).

University of New Brunswick (no date). Available at: https://www.unb.ca/cic/datasets/url-2016.html (Accessed: 6 December 2021).