

Configuration Manual

MSc Research Project
Cybersecurity

Vikas Naidugari
Student ID: x22237020

School of Computing
National College of Ireland

Supervisor: Raza Ul Mustafa

National College of Ireland
MSc Project Submission Sheet
School of Computing



Student Name: Vikas Naidugari
Student ID: x22237020
Programme: MSc. In Cybersecurity **Year:** 2023-2024
Module: Research Project
Lecturer: Raza Ul Mustafa
Submission Due Date: 16 Sep 2024
Project Title: Detecting Real Time Phishing URL's with URL Cleaning Technique
Word Count: 958 **Page Count:** 09

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: Vikas Naidugari

Date: 16 Sep 2024

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	<input type="checkbox"/>
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	<input type="checkbox"/>
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.	<input type="checkbox"/>

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

Vikas Naidugari
Student ID: x22237020

Introduction

The configuration manual will guide you to the hardware and software that was used within the project. The name of the project “Detecting Real Time Phishing URL’s with URL Cleaning Technique”. It further explains the steps that are required to install and reproduce the work. All of the requirements that are necessary for the project are discussed in the following section.

Environmental Setup

The following research lab setup was used to set up the testing lab for the research

- Operating System- Windows 11 Home 64-bit
- Processor: 13th Gen Intel(R) Core(TM) i7 @ 2.60GHz
- Memory: 16.0 GB 3200 MHz
- Graphics – Intel(R) iRIS(Xe) Graphics
- Anaconda Application (Anaconda3, 2024)
- Programming Language – Python
- Environment – Jupyter Notebook

Python Libraries

The following libraries are required to run this application

- Pandas (Fumo, 2017)
- Numpy
- Sklearn (Pedregosa *et al.*, 2011)
- MATPlotlib (*Using Matplotlib — Matplotlib 3.9.1 documentation*, no date).

```

import pandas as pd
from sklearn.preprocessing import LabelEncoder
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, f1_score, precision_score, recall_score
import matplotlib.pyplot as plt
from sklearn.metrics import roc_curve, roc_auc_score
from sklearn.model_selection import cross_val_score

```

Fig 1: Libraries

Implementation and Testing

1. Data Loading

```

import pandas as pd
df=pd.read_csv("C:\\Users\\naidu\\OneDrive\\Thesis Report\\Dataset Phishing.csv")

```

2. Columns in the dataset

```

df.columns

Index(['url', 'length_url', 'length_hostname', 'ip', 'nb_dots', 'nb_hyphens',
      'nb_at', 'nb_qm', 'nb_and', 'nb_or', 'nb_eq', 'nb_underscore',
      'nb_tilde', 'nb_percent', 'nb_slash', 'nb_star', 'nb_colon', 'nb_comma',
      'nb_semicolumn', 'nb_dollar', 'nb_space', 'nb_www', 'nb_com',
      'nb_dslash', 'http_in_path', 'https_token', 'ratio_digits_url',
      'ratio_digits_host', 'punycode', 'port', 'tld_in_path',
      'tld_in_subdomain', 'abnormal_subdomain', 'nb_subdomains',
      'prefix_suffix', 'random_domain', 'shortening_service',
      'path_extension', 'nb_redirection', 'nb_external_redirection',
      'length_words_raw', 'char_repeat', 'shortest_words_raw',
      'shortest_word_host', 'shortest_word_path', 'longest_words_raw',
      'longest_word_host', 'longest_word_path', 'avg_words_raw',
      'avg_word_host', 'avg_word_path', 'phish_hints', 'domain_in_brand',
      'brand_in_subdomain', 'brand_in_path', 'suspicious_tld',
      'statistical_report', 'nb_hyperlinks', 'ratio_intHyperlinks',
      'ratio_extHyperlinks', 'ratio_nullHyperlinks', 'nb_extCSS',
      'ratio_intRedirection', 'ratio_extRedirection', 'ratio_intErrors',
      'ratio_extErrors', 'login_form', 'external_favicon', 'links_in_tags',
      'submit_email', 'ratio_intMedia', 'ratio_extMedia', 'sfh', 'iframe',
      'popup_window', 'safe_anchor', 'onmouseover', 'right_click',
      'empty_title', 'domain_in_title', 'domain_with_copyright',
      'whois_registered_domain', 'domain_registration_length', 'domain_age',
      'web_traffic', 'dns_record', 'google_index', 'page_rank', 'status'],
      dtype='object')

```

3. Handling Categorical Data (encoded status)

Handling the categorical data is very important step because most of the machine learning models does not support the categorical data or sometimes due to the presence of categorical data, predictions made by model can be wrong. So, we are converting the categorical data into numerical form which is present in the status column.

we make the object of Label Encoder import from scikit-learn (Pedregosa *et al.*, 2011) and then we fit it into the status columns and the new numerical values generated by using label encoder is then stored in the new variable named 'encoded status'. This column contains the numerical data (0 for legitimate and 1 for phishing).

```
df['status'].unique()

array(['legitimate', 'phishing'], dtype=object)

from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
df['encoded_status']=le.fit_transform(df['status'])

df.drop(columns='status',inplace=True)

df.head(40)
```

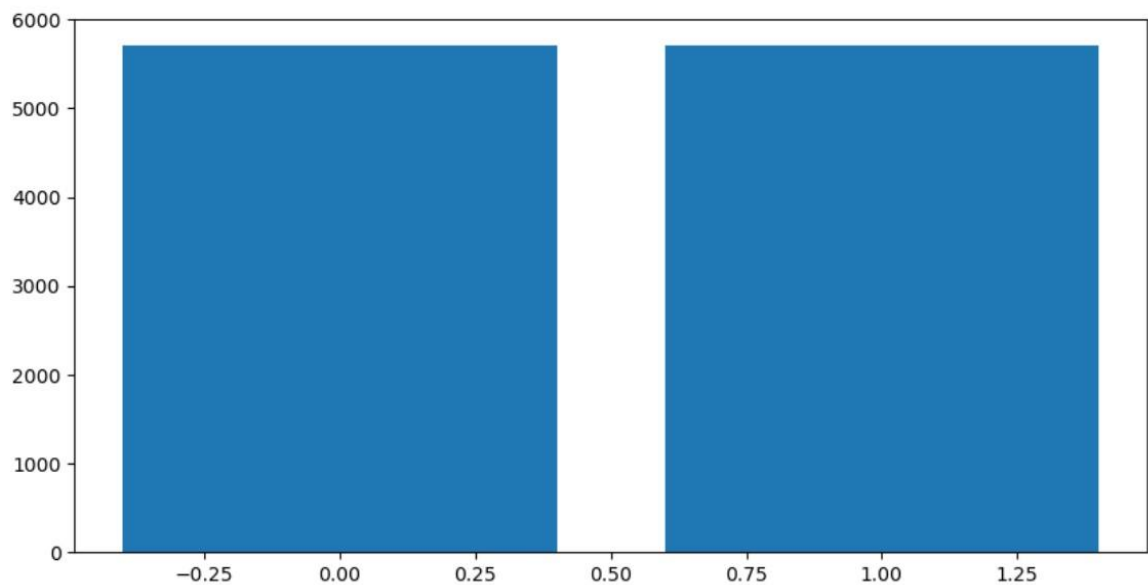
with_copyright	whois_registered_domain	domain_registration_length	domain_age	web_traffic	dns_record	google_index	page_rank	encoded_status
1	0	485	10106	1103	0	0	8	0
0	0	146	2046	0	0	1	0	1
1	0	109	2447	458809	0	0	3	0
1	0	588	7446	383	0	1	8	0
1	0	158	9557	19025	0	0	5	0
1	0	123	3895	817468	0	0	4	0
1	0	1722	2660	46690	0	0	5	0
0	0	235	3783	9280076	0	1	2	1
0	0	374	7295	0	0	0	5	1

4. Exploratory Data Analysis

It is the main step in the process of data analysis and it helps to visualize the patterns, characteristics and relationships between the variables present in the data. Here we can use matplotlib Library (*Using Matplotlib — Matplotlib 3.9.1 documentation*) for visualizing the data.

The bar graph where x-axis represent the unique values in the encoded _status and y-axis contains count of each unique values.

```
import matplotlib.pyplot as plt
x=df['encoded_status'].value_counts()
plt.figure(figsize=(10,5))
plt.bar(x.index,x.values)
plt.show()
```



5. Data Splitting

To get the model with best accuracy and for better performance of the model, we split the data into training and testing set. Where we take 80 percent of the data as training data and remaining 20 percent of data as testing data with random state value of 42. It is very important to split the data into training and testing set to evaluate the performance of the model.

```
X=df.drop(columns='encoded_status')
y=df['encoded_status']
#Splitting the data
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=42)
```

```
categorical_cols = [cname for cname in X_train.columns if X_train[cname].nunique() < 10 and
                    X_train[cname].dtype == "object"]
```

```
numerical_cols = [cname for cname in X_train.columns if X_train[cname].dtype in ['int64', 'float64']]
```

```
my_cols = categorical_cols + numerical_cols
X_train = X_train[my_cols].copy()
X_test = X_test[my_cols].copy()
```

```
from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer

numerical_transformer = SimpleImputer(strategy='constant')

categorical_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='most_frequent')),
    ('onehot', OneHotEncoder(handle_unknown='ignore'))
])

preprocessor = ColumnTransformer(
    transformers=[
        ('num', numerical_transformer, numerical_cols),
        ('cat', categorical_transformer, categorical_cols)
    ])
```


6. Feature Scaling

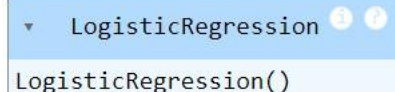
Many machine learning algorithms requires the data scaling for producing good results. So, it is important that all the features which are given as the input to the model is in a particular range. So, in below code we standardize the features using StandardScaler. And fit it on X_train and X_test to standardize it.

```
#Feature scaling
from sklearn.preprocessing import StandardScaler
ss=StandardScaler()
X_train=ss.fit_transform(X_train)
X_test=ss.transform(X_test)
```

7. Machine Learning Algorithms

we imported the modelling from scikit-learn library and then we created its object named lr. After this we fit it in the X_train and y_train for training purpose.

```
#Using Logistic Regression
from sklearn.linear_model import LogisticRegression
lr=LogisticRegression()
lr.fit(X_train,y_train)
```



```
LogisticRegression()
```

We evaluated many machine learning models to determine if websites are phishing or real. Tests included Decision Tree, Random Forest, K-Nearest Neighbors (KNN), Support Vector Machine (SVM), Logistic Regression, and Naïve Bayes. The Random Forest classifier outperformed the rest in terms of accuracy, successfully differentiating between reputable and fraudulent websites.

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
knc = KNeighborsClassifier()
knc.fit(X_train, y_train)
nb = GaussianNB()
nb.fit(X_train, y_train)
rfc = RandomForestClassifier()
rfc.fit(X_train, y_train)
dt = DecisionTreeClassifier()
dt.fit(X_train, y_train)
svc = SVC()
svc.fit(X_train, y_train)
```

8. Cross – Validation Process

In this step we will find the accuracy,f1, recall and precision scores. Through this results, we will get to know about the performance of model.

```
from sklearn.metrics import accuracy_score, f1_score, precision_score, recall_score
```

```
# Define models
models = [
    ('Logistic Regression', LogisticRegression()),
    ('Support Vector Machine', SVC()),
    ('KNeighbors', KNeighborsClassifier()),
    ('Random Forest', RandomForestClassifier()),
    ('Decision Tree', DecisionTreeClassifier()),
    ('Naive Bayes', GaussianNB())
]
```

```
# Perform cross-validation and print results
for name, model in models:
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)

    accuracy = accuracy_score(y_test, y_pred)
    F1_Score = f1_score(y_test, y_pred)
    precision = precision_score(y_test, y_pred)
    recall = recall_score(y_test, y_pred)

    print(f"\x1B[4m{name}\x1B[0m")
    print("Accuracy Score :", accuracy)
    print("F1 Score :", F1_Score)
    print("Precision:", precision)
    print("Recall:", recall)
    print("-----")
```

Logistic Regression

Accuracy Score : 0.9483814523184602
F1 Score : 0.9486510008703221

Logistic Regression

Accuracy Score : 0.9483814523184602
F1 Score : 0.9486510008703221
Precision: 0.9503051438535309
Recall: 0.947002606429192

Support Vector Machine

Accuracy Score : 0.9593175853018373
F1 Score : 0.9596529284164859
Precision: 0.9584055459272097
Recall: 0.9609035621198957

KNeighbors

Accuracy Score : 0.9413823272090989
F1 Score : 0.9408127208480566
Precision: 0.9568733153638814
Recall: 0.9252823631624674

Random Forest

Accuracy Score : 0.9650043744531933
F1 Score : 0.965034965034965
Precision: 0.9709762532981531
Recall: 0.9591659426585578

Decision Tree

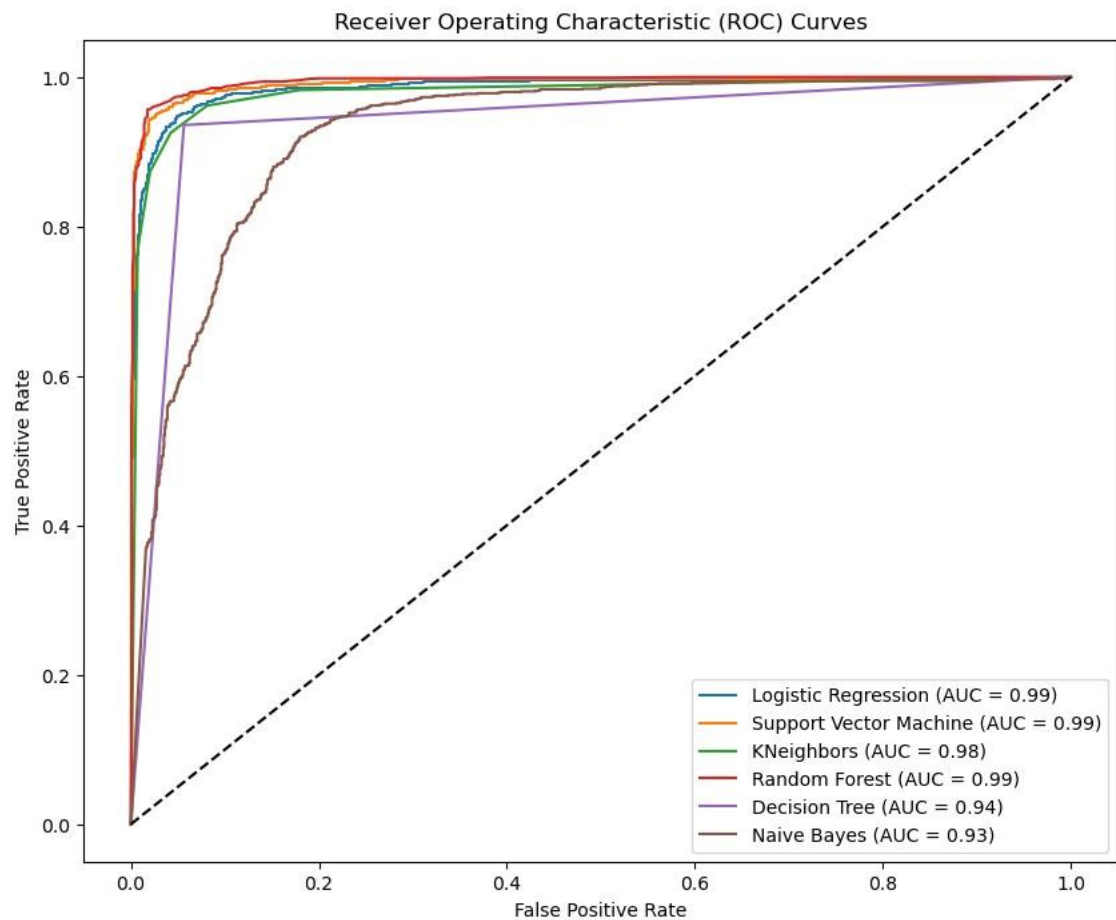
Accuracy Score : 0.9339457567804025
F1 Score : 0.9345470307758994
Precision: 0.9325259515570934
Recall: 0.9365768896611643

Naive Bayes

Accuracy Score : 0.6881014873140857
F1 Score : 0.5655088360755637
Precision: 0.9469387755102041
Recall: 0.40312771503040834

9. Receiver Operating Characteristics (ROC) Curve

It is used for predicting the classification thresholds. A ROC curve is a graph that is used for showing the performance of classification model at all thresholds. This curve plots two parameters True Positive Ratio and False Positive Ratio.



References

Fumo, J. (2017) 'Pandas Library in a Nutshell — Intro To Machine Learning #3', *Simple AI*, 29 January. Available at: <https://medium.com/simple-ai/pandas-library-in-a-nutshell-intro-to-machine-learning-3-acbd39ec5c9c> (Accessed: 12 August 2024).

Pedregosa, F. *et al.* (2011) 'Scikit-learn: Machine Learning in Python', *J. Mach. Learn. Res.*, 12(null), pp. 2825–2830.

Using Matplotlib — Matplotlib 3.9.1 documentation (no date). Available at: <https://matplotlib.org/stable/users/index.html> (Accessed: 12 August 2024).