National College of Ireland

Configuration Manual

MSc Data Analytics
Research Project

Preetham Madeti
22142258

School of Computing
National College of Ireland

Supervisor: Abdul Qayum

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Preetham Madeti		
Student ID:	22142258		
Programme:	MSc Data Analytics	**Year:**	2023/2024
Module:	Research Project		
Supervisor:	Mr. Abdul Qayum		
Submission Due Date:	31/01/2024		
Project Title:	Implementing a Hybrid System for Accurately Detecting Phishing URLs with Machine Learning and Deep Learning Techniques.		
Word Count:	735	**Page Count:**	10

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: Preetham Madeti

Date: 29/01/2024

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

Preetham Madeti
22142258

Introduction:

This manual provides detailed instructions for configuring and deploying the phishing URL detection system developed in this research project. The system employs a hybrid model integrating machine learning and deep learning techniques to accurately identify phishing URLs.

1 System Requirements:

To guarantee efficient model processing and to minimize the duration required, it's crucial to be equipped with the necessary hardware and software resources.

1.1. Hardware Requirements:

The implementation is performed on an HP Pavilion; the configuration of the device is as follows.

1. Processor: Intel(R) Core (TM) i5-9300H CPU @ 2.40GHz
2. RAM: 8.00 GB (7.84 GB usable)
3. Hard Disk: 256GB SSD, 1 TB HDD
4. OS Windows 10 Pro 64 – bit

1.2 Software Requirements:

Before beginning the model construction phase, the below mentioned software, libraries, and tools were set up and installed on the system.

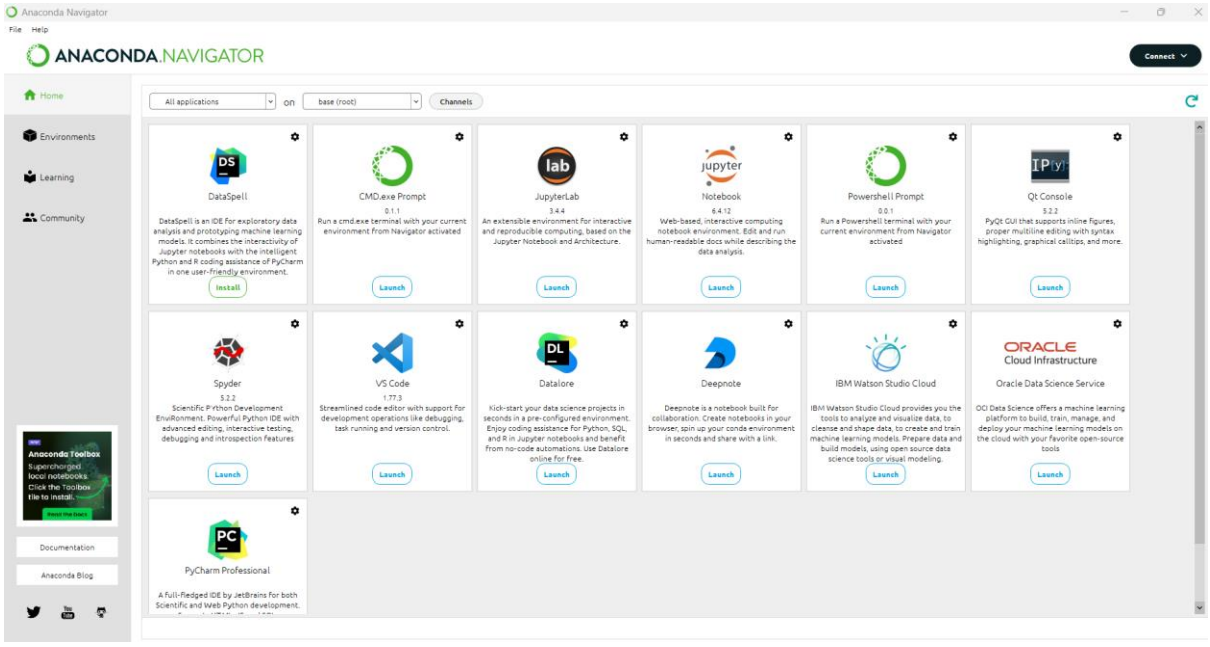
Software/Tools	Version	Information
Python		To develop the model python is used in this project.
Anaconda		Anaconda stands as a highly favoured platform for the data science community, offering capabilities for computational work, managing libraries, and deploying models, all within a Windows-friendly environment
Pandas		It is particularly well-suited for dealing with tabular data, such as data stored in spreadsheets or databases.

NumPy		NumPy, an open-source tool from 2023, is utilized for tackling intricate mathematical issues within data.
Sci-kit Learn		This library is employed for tasks like Classification, Regression, and data preprocessing. (Scikit-learn: Machine Learning in Python — scikit-learn 0.24.2 documentation, 2023).

2. Implementation:

In this section there is a complete guide to run the project in any windows system.

1. Download and Install Anaconda Software in the windows system. (https://www.anaconda.com/products/individual)



2. Open the Jupyter Notebook from Anaconda.

3. After opening jupyter notebook click on the new notebook (python 3).
4. In notebook, Import all the required libraries.

```
import matplotlib.pyplot as plt
import seaborn as sns
from wordcloud import WordCloud
from urllib.parse import urlparse, parse_qs
from collections import Counter
```

```
from sklearn.metrics import classification_report as urlphish_hyrder_E1
from sklearn.metrics import confusion_matrix as urlphish_hyrder_E2
from sklearn.metrics import ConfusionMatrixDisplay as urlphish_hyrder_E3
import time as urlphish_hyrder_E4
from sklearn.model_selection import GridSearchCV as urlphish_hyrder_E5
```

```
from sklearn.metrics import classification_report as urlphish_hyrder_E1
from sklearn.metrics import confusion_matrix as urlphish_hyrder_E2
from sklearn.metrics import ConfusionMatrixDisplay as urlphish_hyrder_E3
import time as urlphish_hyrder_E4
from sklearn.model_selection import GridSearchCV as urlphish_hyrder_E5
from sklearn.ensemble import AdaBoostClassifier as urlphish_hyrder_E6
from sklearn.ensemble import RandomForestClassifier as urlphish_hyrder_E7
from sklearn.naive_bayes import GaussianNB as urlphish_hyrder_E8
from sklearn.tree import DecisionTreeClassifier as urlphish_hyrder_E9
from sklearn.neural_network import MLPClassifier as urlphish_hyrder_E10
from sklearn.ensemble import VotingClassifier as urlphish_hyrder_E11
```

5. Import the Provided Dataset.

```
urlphish_hyrd = urlphish_hyrde.read_csv('phishing_site_urls.csv')
```

6. Next Step will be Pre Processing Step will be performed using following Code.

```
]: urlphish_hyrd.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 549346 entries, 0 to 549345
Data columns (total 2 columns):
 #   Column  Non-Null Count   Dtype
---  ------  --------------   -----
 0   URL     549346 non-null  object
 1   Label   549346 non-null  object
dtypes: object(2)
memory usage: 8.4+ MB
```

pick null data out

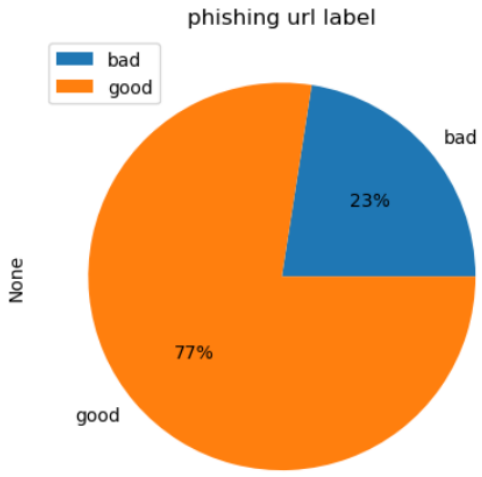
```
]: urlphish_hyrd.isna().any()

]: URL      False
   Label    False
   dtype: bool
```

pick duplicate data out

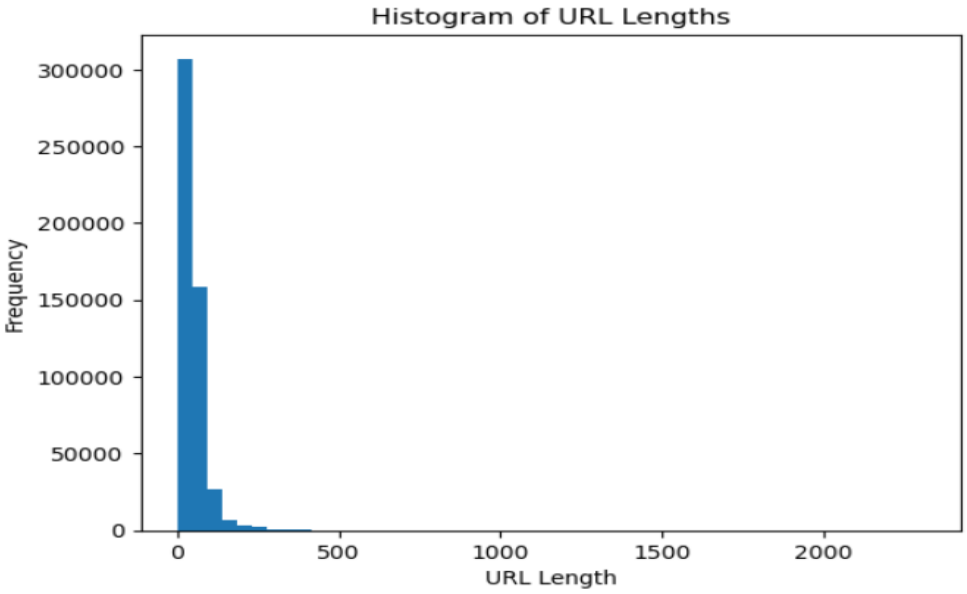
```
]: urlphish_hyrd.duplicated()

]: 0         False
   1         False
   2         False
   3         False
   4         False
             ...
   549341     True
   549342     True
   549343     True
   549344     True
   549345     True
   Length: 549346, dtype: bool
```

6. Exploratory Data Analysis has been Performed and Visualisation has been done using following Code

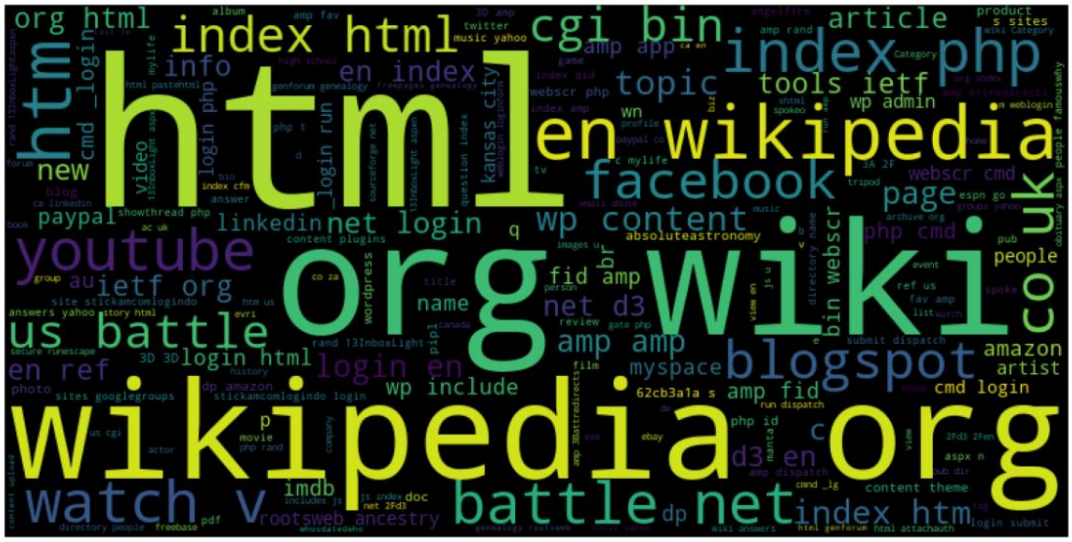
```
: urlphish_hyrd.groupby('Label').size().plot(kind='pie', legend=True, autopct='%1.0f%%', title='phishing url label')
: <AxesSubplot:title={'center':'phishing url label'}, ylabel='None'>
```



```
urlphish_hyrd['url_length'] = urlphish_hyrd['URL'].apply(len)
plt.hist(urlphish_hyrd['url_length'], bins=50)
plt.title('Histogram of URL Lengths')
plt.xlabel('URL Length')
plt.ylabel('Frequency')
plt.show()
```



In [17]:
```
text = ' '.join(urlphish_hyrd['URL'])
wordcloud = WordCloud(width=800, height=400).generate(text)
plt.figure(figsize=(15, 10))
plt.imshow(wordcloud)
plt.axis('off')
plt.show()
```



7. After Data Pre Processing the Data Splitting is Performed before Building a Model

In [11]:
```
from sklearn.model_selection import train_test_split as urlphish_hyrderios

hish_hyrd_R = 99
hish_hyrd_Sa =0.4
#..... train= 60% ...........
inputN_hybrid, inputS_hybrid, outputN_hybrid, outputS_hybrid = urlphish_hyrderios(input_hybrid, output_hybrid, test_size=hish_hyr
hish_hyrd_Sb =0.5
#.....  test= 20% ,validation= 20% ............
inputV_hybrid, inputS_hybrid, outputV_hybrid, outputS_hybrid = urlphish_hyrderios(inputS_hybrid, outputS_hybrid, test_size=hish_h


print(inputN_hybrid.shape)
print(inputV_hybrid.shape)
print(inputS_hybrid.shape)
```

8. Hybrid Models Implementation has been Performed with the following Code

In [16]:
```
urlphish_vrt = {'voting': ['soft', 'hard']}

urlphish_hyrder_m1 = urlphish_hyrder_E7(criterion= 'entropy', n_estimators= 20, n_jobs= 10)

urlphish_hyrder_m2 = urlphish_hyrder_E10(activation= 'relu', learning_rate= 'constant', solver= 'adam')

urlphish_vrtr = urlphish_hyrder_E11(estimators=[('randomforest', urlphish_hyrder_m1), ('mlp', urlphish_hyrder_m2)])

urlphish_vrtr = urlphish_hyrder_E5(urlphish_vrtr, urlphish_vrt, cv=2)
urlphish_vrtr.fit(inputN_hybrid[:1000], outputN_hybrid[:1000])

print(urlphish_vrtr.best_params_)
print("score_value : ", urlphish_vrtr.best_score_)
```

```
{'voting': 'soft'}
score_value :  0.801
```

In [17]:
```
urlphish_hyrder_a = urlphish_hyrder_E4.time()
hyrder_A = urlphish_hyrder_E11(estimators=[('randomforest', urlphish_hyrder_m1), ('mlp', urlphish_hyrder_m2)], **urlphish_vrtr.be
hyrder_A.fit(inputN_hybrid, outputN_hybrid)

urlphish_hyrder_b = urlphish_hyrder_E4.time()
print("\n Training------time :", urlphish_hyrder_b-urlphish_hyrder_a,"\n")

urlphish_hyrder_a = urlphish_hyrder_E4.time()

P_hybrid = hyrder_A.predict(inputV_hybrid)
print(urlphish_hyrder_E1(outputV_hybrid, P_hybrid))
E2 = urlphish_hyrder_E2(outputV_hybrid, P_hybrid)
E3 = urlphish_hyrder_E3(confusion_matrix = E2, display_labels = [0, 1])
E3.plot()

urlphish_hyrder_b = urlphish_hyrder_E4.time()
print("\n validation--------time  :", urlphish_hyrder_b-urlphish_hyrder_a,"\n")
```

9. The Accuracy is considered as evaluation factor after Model Implementation

```
urlphish_hyrder_a = urlphish_hyrder_E4.time()
hyrder_A = urlphish_hyrder_E11(estimators=[('randomforest', urlphish_hyrder_m1), ('mlp', urlphish_hyrder_m2)], **urlphish_vrtr.be
hyrder_A.fit(inputN_hybrid, outputN_hybrid)

urlphish_hyrder_b = urlphish_hyrder_E4.time()
print("\n Training------time :", urlphish_hyrder_b-urlphish_hyrder_a,"\n")

urlphish_hyrder_a = urlphish_hyrder_E4.time()

P_hybrid = hyrder_A.predict(inputV_hybrid)
print(urlphish_hyrder_E1(outputV_hybrid, P_hybrid))
E2 = urlphish_hyrder_E2(outputV_hybrid, P_hybrid)
E3 = urlphish_hyrder_E3(confusion_matrix = E2, display_labels = [0, 1])
E3.plot()

urlphish_hyrder_b = urlphish_hyrder_E4.time()
print("\n validation--------time  :", urlphish_hyrder_b-urlphish_hyrder_a,"\n")
```

```
 Training------time : 768.395833492279

              precision    recall  f1-score   support

           0       0.87      0.55      0.67     22821
           1       0.88      0.98      0.93     78618

    accuracy                           0.88    101439
   macro avg       0.88      0.76      0.80    101439
weighted avg       0.88      0.88      0.87    101439
```

In [18]:
```
urlphish_hyrder_a = urlphish_hyrder_E4.time()

P_hybrid = hyrder_A.predict(inputS_hybrid)
print(urlphish_hyrder_E1(outputS_hybrid, P_hybrid))
E2 = urlphish_hyrder_E2(outputS_hybrid, P_hybrid)
E3 = urlphish_hyrder_E3(confusion_matrix = E2, display_labels = [0, 1])
E3.plot()

urlphish_hyrder_b = urlphish_hyrder_E4.time()
print("\n testing--------time  :", urlphish_hyrder_b-urlphish_hyrder_a,"\n")
```

```
              precision    recall  f1-score   support

           0       0.87      0.55      0.68     22845
           1       0.88      0.98      0.93     78595

    accuracy                           0.88    101440
   macro avg       0.88      0.76      0.80    101440
weighted avg       0.88      0.88      0.87    101440


 testing--------time  : 0.5752835273742676
```

Total Execution Time:

• The overall duration taken to train the dynamic model was 768.395 seconds, while the time taken to test the data using the trained model amounted to 0.575 milliseconds.

The concluding code files are included within three distinct project files, titled 'Phishing URL Data Cleaning.ipynb', 'Implementing ML-DL Models.ipynb', and 'Hybrid Model with ML-DL Models.ipynb', respectively.

References

Anaconda. 2021. Anaconda | The World's Most Popular Data Science Platform. [online] Available at: .<https://www.anaconda.com/>.

Numpy.org. 2021. NumPy. [online] Available at: <https://numpy.org/>.

Scikit-learn.org. 2021.

scikit-learn machine learning in Python — scikit-learn 0.24.2

documentation. [online] Available at: <https://scikit-learn.org/stable/>.