Configuration Manual

This document provides a comprehensive guide on using software and tools for project execution, including detailed instructions for software installation and a systematic approach to project success. The report details the specific hardware requirements needed to efficiently run Machine Learning algorithms.

1. Hardware Specifications

Hardware Component	Requirements
Processor (CPU)	HP Pavilion
Memory (RAM)	12th Gen Intel(R) Core(TM) i5-1235U 1.30 GHz
Network Connection	Stable internet connection
Operating System	Compatible with various OS
System type	512 SSD
ML Model (if applicable)	HP Pavilion

2. Software Specifications

Software Component	Requirements
Python	Python 3.10.12
Flask	Flask web framework
NumPy	Numerical computing library
pandas	Data manipulation library
scikit-learn	Machine learning library
mlxtend	Machine learning library
Requests	HTTP library for making requests
URLFeatureExtraction	Custom Python module (dependencies may vary)
pickle	Python object serialization library
smtplib	SMTP email library for sending emails
A web browser	For accessing and testing the web application
An operating system	Compatible with the required software
Visual studio version	Version: 1.85.0 OS: Windows_NT x64 10.0.22621

Software Component	Requirements
Google Colab	3.10.12

3. Python Packages and Imports

1. **Flask Packages:**

   - **Flask:** The Flask web framework is the core of the application, used for routing and handling HTTP requests.

   - **request:** Routing and handling HTTP requests are handled by the Flask web framework, which is the core of the application.

   - **render_template:** Used to render HTML templates for web pages.

2. **Data Manipulation and Machine Learning Packages:**

   - **numpy and pandas:** Used for data manipulation and handling.**(*Introduction to NumPy*, n.d.)**

   - **pickle:** Used for loading a pre-trained machine learning model from a saved file.**(*Libraries in Python - GeeksforGeeks*, n.d.)**

   - **sklearn:** Scikit-learn, a machine learning library, is used for making predictions with the loaded model.**(*Libraries in Python - GeeksforGeeks*, n.d.)**

   - **mlxtend:** An extension library for scikit-learn that may contain additional functionality for machine learning.**(*Libraries in Python - GeeksforGeeks*, n.d.)**

3. **Email Handling:**

   - **email.mime:** A module that facilitates the creation and management of email messages

- **smtplib:** Used for sending email messages via the Simple Mail Transfer Protocol (SMTP).

4. **Custom Modules:**

   - **urlfeatureextraction:** A custom Python module that is imported to perform URL feature extraction. The specific dependencies for this module may vary and should be installed separately.

5. **Other General Imports:**

   - Various Python modules and functions for general functionality within the code, such as string manipulation and handling data structures.

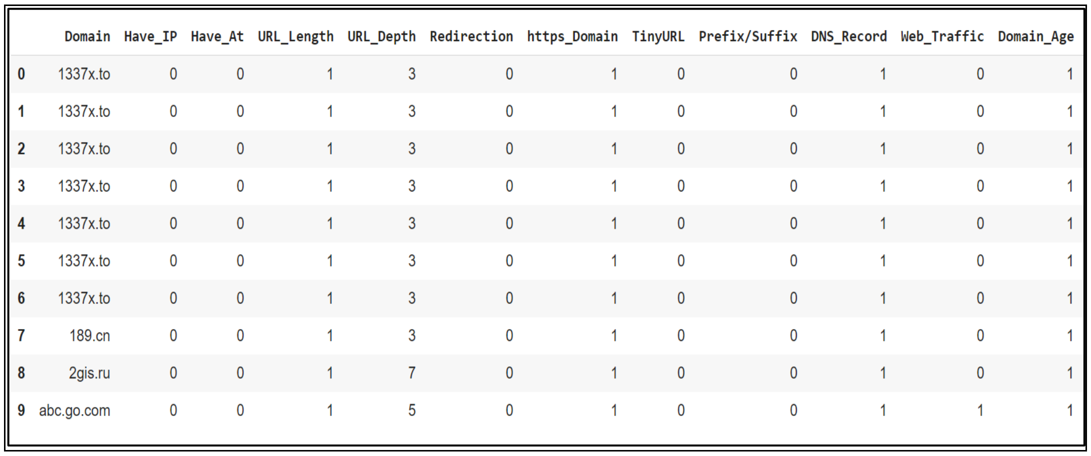
4. Dataset Overview and Data Loading

This study aims to tackle online criminal activities, with particular emphasis on URLs. Our approach to categorizing malicious URLs into five types is lightweight. Over 45,000 instances are included in the dataset, which provides a comprehensive understanding of online threats. We scrutinize the obfuscation techniques employed by hackers. Our research is aided by this dataset, which enables us to explore online security effectively.

Data Loading

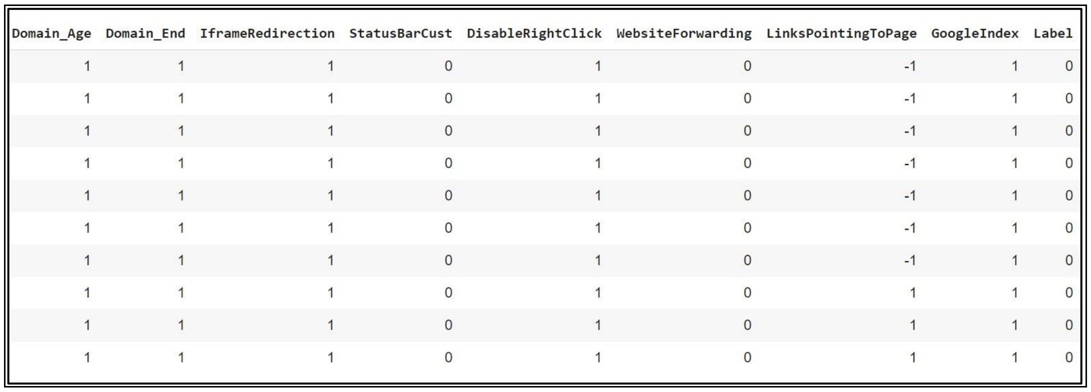
The provided code snippet demonstrates data loading using the Pandas library. It displays the first 10 rows of the loaded data using the **head** method. This operation allows for quick exploration and understanding of the dataset's structure and content.

```
#data loading
data =
pd.read_csv('/content/drive/MyDrive/phishing_url_classification/Data/final_dataframe.cs
v')
data.head(10)
```



	Domain	Have_IP	Have_At	URL_Length	URL_Depth	Redirection	https_Domain	TinyURL	Prefix/Suffix	DNS_Record	Web_Traffic	Domain_Age
0	1337x.to	0	0	1	3	0	1	0	0	1	0	1
1	1337x.to	0	0	1	3	0	1	0	0	1	0	1
2	1337x.to	0	0	1	3	0	1	0	0	1	0	1
3	1337x.to	0	0	1	3	0	1	0	0	1	0	1
4	1337x.to	0	0	1	3	0	1	0	0	1	0	1
5	1337x.to	0	0	1	3	0	1	0	0	1	0	1
6	1337x.to	0	0	1	3	0	1	0	0	1	0	1
7	189.cn	0	0	1	3	0	1	0	0	1	0	1
8	2gis.ru	0	0	1	7	0	1	0	0	1	0	1
9	abc.go.com	0	0	1	5	0	1	0	0	1	1	1

Figure 1 Output of the dataset loading



Domain_Age	Domain_End	IframeRedirection	StatusBarCust	DisableRightClick	WebsiteForwarding	LinksPointingToPage	GoogleIndex	Label
1	1	1	0	1	0	-1	1	0
1	1	1	0	1	0	-1	1	0
1	1	1	0	1	0	-1	1	0
1	1	1	0	1	0	-1	1	0
1	1	1	0	1	0	-1	1	0
1	1	1	0	1	0	-1	1	0
1	1	1	0	1	0	-1	1	0
1	1	1	0	1	0	1	1	0
1	1	1	0	1	0	1	1	0
1	1	1	0	1	0	1	1	0

Figure 2 output of dataset loading

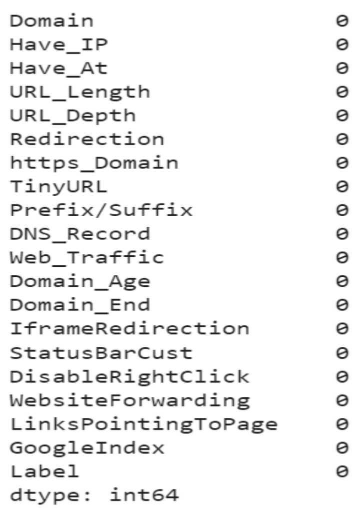
5. Data Cleaning and Preprocessing

To prepare a dataset for analysis and modeling, data cleaning and preprocessing are necessary tasks. The task involves handling missing values, eliminating duplicates, transforming and encoding data, addressing outliers, feature engineering, and separating the dataset for evaluation. Data quality and compatibility with machine learning algorithms are ensured by the specific tasks that depend on the dataset's nature and objectives.

```
#checking for null values
data.isna().sum()
```

The provided code snippet checks for and counts the number of null (missing) values in the dataset 'data' using the **isna().sum()** method. This operation helps identify the extent of missing data in the dataset, which is a crucial step in data quality assessment and data preprocessing

A count of null values in each column of the 'data' dataset is generated by the 'data.isna().sum()' code scribble. Identifying columns with missing data is crucial, and this step can help make decisions about handling null values, such as imputation or removal, during data preprocessing.

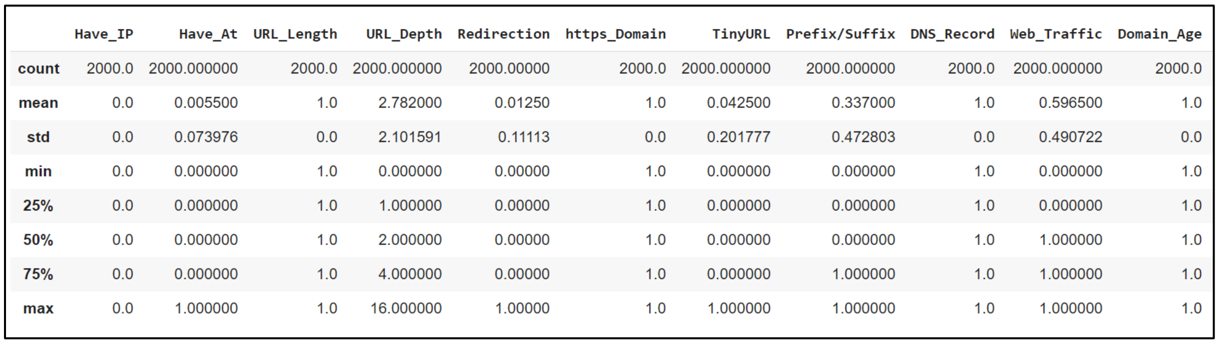


```
Domain                 0
Have_IP                0
Have_At                0
URL_Length             0
URL_Depth              0
Redirection            0
https_Domain           0
TinyURL                0
Prefix/Suffix          0
DNS_Record             0
Web_Traffic            0
Domain_Age             0
Domain_End             0
IframeRedirection      0
StatusBarCust          0
DisableRightClick      0
WebsiteForwarding      0
LinksPointingToPage    0
GoogleIndex            0
Label                  0
dtype: int64
```

Figure 3Output of data cleaning and preprocessing

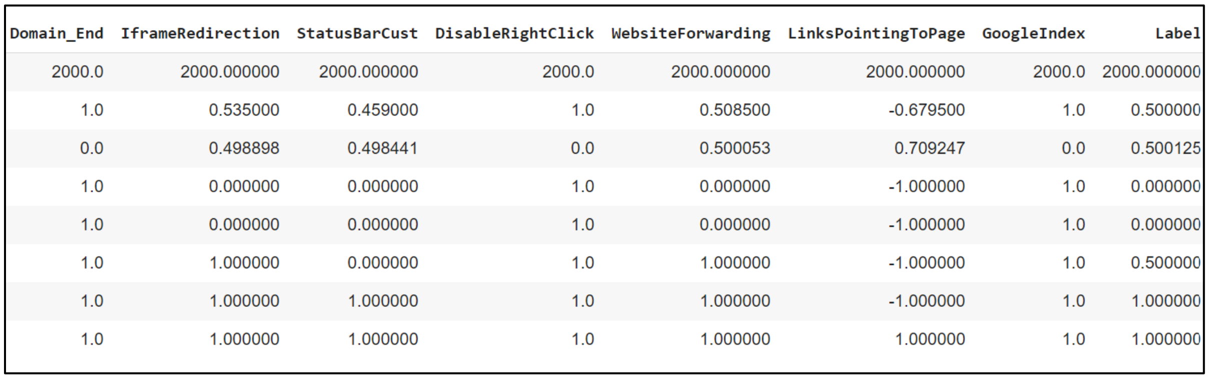
```
#statistic of data
data.describe()
```

Pandas are utilized in this code, which has the comment '#statistic of data', to generate descriptive statistics for a dataset. 'data.description()' computes important statistical metrics for numerical columns, including count, mean, and percentiles. The statistical data provides a quick overview of the dataset's numerical features, which aids in data exploration and analysis.



	Have_IP	Have_At	URL_Length	URL_Depth	Redirection	https_Domain	TinyURL	Prefix/Suffix	DNS_Record	Web_Traffic	Domain_Age
count	2000.0	2000.000000	2000.0	2000.000000	2000.00000	2000.0	2000.000000	2000.000000	2000.0	2000.000000	2000.0
mean	0.0	0.005500	1.0	2.782000	0.01250	1.0	0.042500	0.337000	1.0	0.596500	1.0
std	0.0	0.073976	0.0	2.101591	0.11113	0.0	0.201777	0.472803	0.0	0.490722	0.0
min	0.0	0.000000	1.0	0.000000	0.00000	1.0	0.000000	0.000000	1.0	0.000000	1.0
25%	0.0	0.000000	1.0	1.000000	0.00000	1.0	0.000000	0.000000	1.0	0.000000	1.0
50%	0.0	0.000000	1.0	2.000000	0.00000	1.0	0.000000	0.000000	1.0	1.000000	1.0
75%	0.0	0.000000	1.0	4.000000	0.00000	1.0	0.000000	1.000000	1.0	1.000000	1.0
max	0.0	1.000000	1.0	16.000000	1.00000	1.0	1.000000	1.000000	1.0	1.000000	1.0

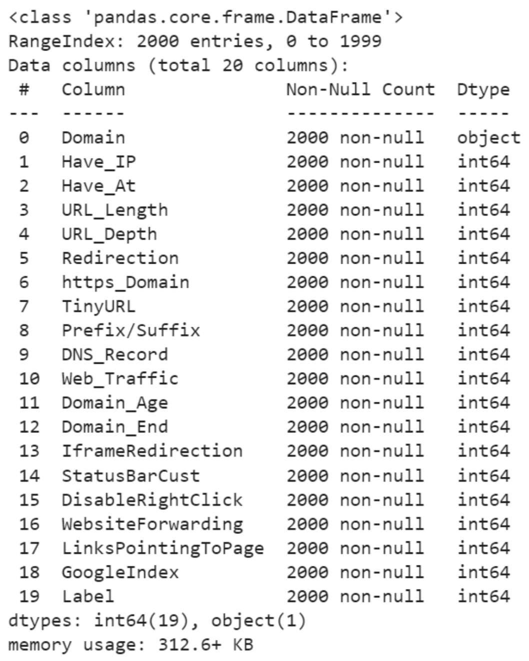
Figure 4 Output of the statistical data



Domain_End	IframeRedirection	StatusBarCust	DisableRightClick	WebsiteForwarding	LinksPointingToPage	GoogleIndex	Label
2000.0	2000.000000	2000.000000	2000.0	2000.000000	2000.000000	2000.0	2000.000000
1.0	0.535000	0.459000	1.0	0.508500	-0.679500	1.0	0.500000
0.0	0.498898	0.498441	0.0	0.500053	0.709247	0.0	0.500125
1.0	0.000000	0.000000	1.0	0.000000	-1.000000	1.0	0.000000
1.0	0.000000	0.000000	1.0	0.000000	-1.000000	1.0	0.000000
1.0	1.000000	0.000000	1.0	1.000000	-1.000000	1.0	0.500000
1.0	1.000000	1.000000	1.0	1.000000	-1.000000	1.0	1.000000
1.0	1.000000	1.000000	1.0	1.000000	1.000000	1.0	1.000000

Figure 5 Output of the statistical data

```
#data information
data.info()
```



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2000 entries, 0 to 1999
Data columns (total 20 columns):
 #   Column               Non-Null Count  Dtype 
---  ------               --------------  ----- 
 0   Domain               2000 non-null   object
 1   Have_IP              2000 non-null   int64 
 2   Have_At              2000 non-null   int64 
 3   URL_Length           2000 non-null   int64 
 4   URL_Depth            2000 non-null   int64 
 5   Redirection          2000 non-null   int64 
 6   https_Domain         2000 non-null   int64 
 7   TinyURL              2000 non-null   int64 
 8   Prefix/Suffix        2000 non-null   int64 
 9   DNS_Record           2000 non-null   int64 
 10  Web_Traffic          2000 non-null   int64 
 11  Domain_Age           2000 non-null   int64 
 12  Domain_End           2000 non-null   int64 
 13  IframeRedirection    2000 non-null   int64 
 14  StatusBarCust        2000 non-null   int64 
 15  DisableRightClick    2000 non-null   int64 
 16  WebsiteForwarding    2000 non-null   int64 
 17  LinksPointingToPage  2000 non-null   int64 
 18  GoogleIndex          2000 non-null   int64 
 19  Label                2000 non-null   int64 
dtypes: int64(19), object(1)
memory usage: 312.6+ KB
```

Figure 6 Output of data information

```
#pie chart of target class(Label)
df1 =
data['Label'].value_counts().reset_index().rename(columns={'index':'Label','Label':'count'
})
fig = px.pie(df1, values='count', names='Label', title='count plot of target class(Label)')
fig.show()
```

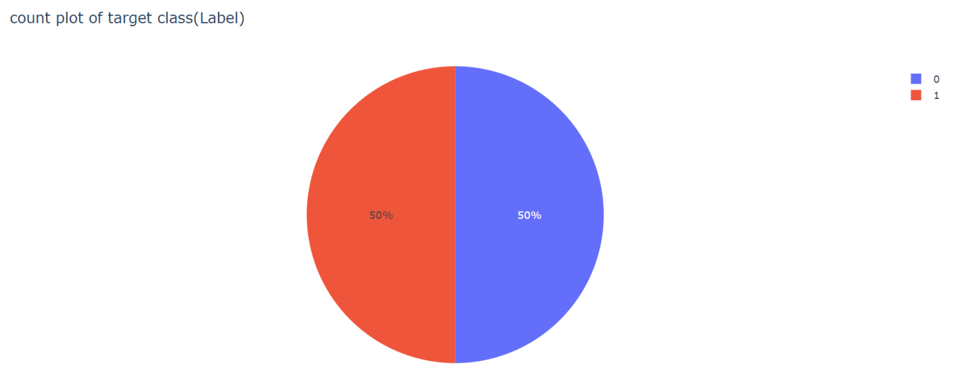
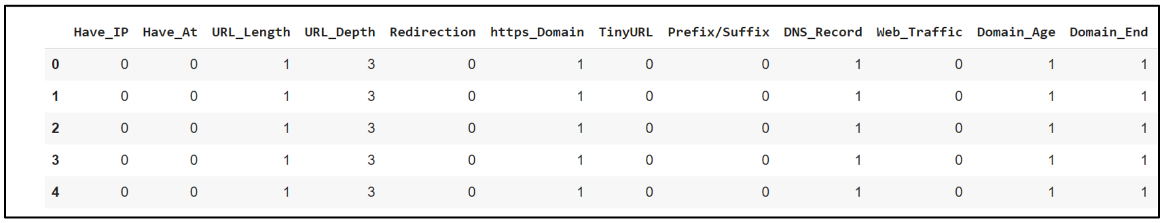


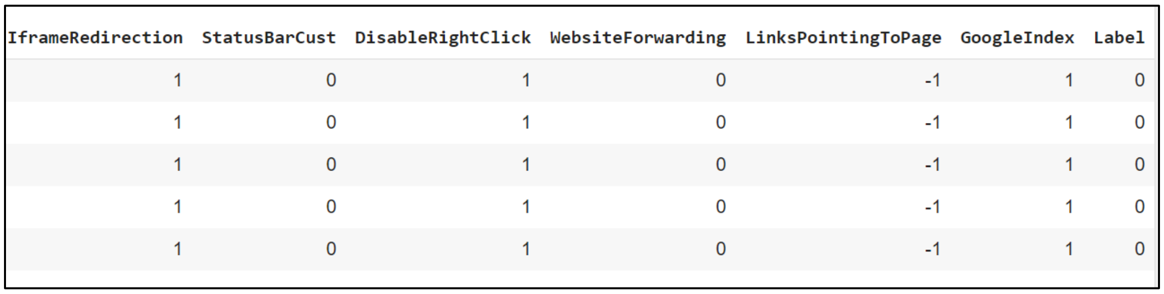
Figure 7 Output of visualization pie chart of phishing and legitimate URLs

This code, indicated by the comment '#dropping unnecessary columns,' uses Pandas to remove the 'Domain' column from the dataset using the 'data.drop()' function. By setting the 'axis' argument to 'columns' and setting 'inplace=True', the change will be applied directly to the current dataset. When certain columns are deemed unnecessary for analysis or model training, this step is useful. The first five rows of the updated dataset can be seen using 'data.head(5)', which gives a glimpse of the data without the 'Domain' column.



	Have_IP	Have_At	URL_Length	URL_Depth	Redirection	https_Domain	TinyURL	Prefix/Suffix	DNS_Record	Web_Traffic	Domain_Age	Domain_End
0	0	0	1	3	0	1	0	0	1	0	1	1
1	0	0	1	3	0	1	0	0	1	0	1	1
2	0	0	1	3	0	1	0	0	1	0	1	1
3	0	0	1	3	0	1	0	0	1	0	1	1
4	0	0	1	3	0	1	0	0	1	0	1	1

Figure 8 Output of featureextration



IframeRedirection	StatusBarCust	DisableRightClick	WebsiteForwarding	LinksPointingToPage	GoogleIndex	Label
1	0	1	0	-1	1	0
1	0	1	0	-1	1	0
1	0	1	0	-1	1	0
1	0	1	0	-1	1	0
1	0	1	0	-1	1	0

Figure 9 Output of feature extraction

Data correlation

```
#correlation matrix
correlation = data.corr().round(2)
plt.figure(figsize = (10,7))
sns.heatmap(correlation, annot = True, cmap='BuPu')
```

This Python code, using Pandas, Matplotlib, and Seaborn, generates and visualizes a correlation matrix for a dataset. It calculates correlations between columns with 'data.corr()' and displays the matrix as a heatmap. The heatmap includes annotated values, and color mapping helps visualize positive correlations in lighter shades and negative correlations in darker shades.

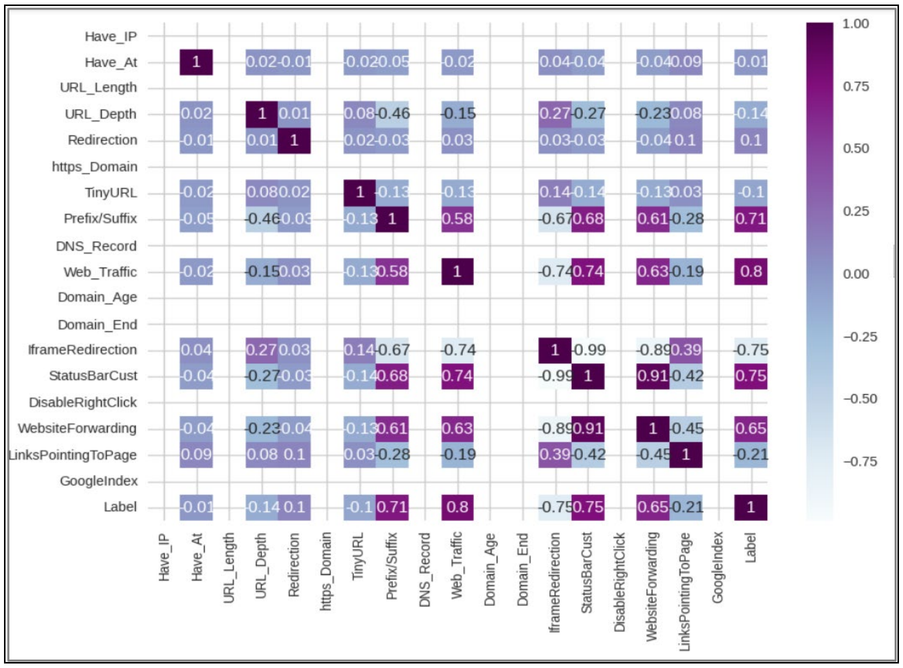


Figure 10 Output of statistic correlation

Data splitting

```
#splitting data into X and y.
X = data.drop('Label',axis='columns')
Y = data['Label']
```

This Python code splits the dataset into features (X) and the target variable (Y) for machine learning. 'X' contains the features, excluding the 'Label' column, while 'Y' holds the 'Label' column as the target variable. This separation sets up the data for subsequent machine learning tasks, clarifying the distinction between features and the target.

```
#splitting data into train, test
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.1, stratify=Y)
```

This Python code snippet performs two key tasks in machine learning. First, it splits the dataset into training and testing sets using 'train_test_split' from scikit-learn, crucial for model evaluation and generalization assessment. Second, it conducts feature importance analysis with the Extra Trees Classifier, helping identify significant features for model interpretation and refinement.

6. Machine Learning Algorithms

1. Logistic regression

This code segment establishes a list called 'select_from_model_list' that contains feature names in order of importance, with the most significant ones at the top. All feature names are stored in feature_name, and indices ensure that they are in the correct order. The Extra Trees Classifier's influential features are represented in a user-friendly way in this list. 'select_from_model_list' is an invaluable tool for further analysis or feature selection, which improves model interpretability and optimization.

```
log_model = LogisticRegression(max_iter=4)
logit = log_model
logit.fit(X_train,y_train)
y_pred = logit.predict(X_test)
```

```
#accuracy score
print("Accuracy Score: ",accuracy_score(y_test,y_pred))
```

Output: Accuracy Score: 0.825

The code is responsible for calculating and displaying the accuracy score of a logistic regression model. The actual labels ('y_test') and the predicted labels ('y_pred') from the model are compared using the 'accuracy_score' function of' sklearn.metrics'. The accuracy score indicates the proportion of correctly classified instances in the test dataset, providing an assessment of the model's predictive performance.

2. Confusion matrix

```
#confusion Matrix
matrix =confusion_matrix(y_test, y_pred)
class_names=[0,1]
fig, ax = plt.subplots()
tick_marks = np.arange(len(class_names))
plt.xticks(tick_marks, class_names)
plt.yticks(tick_marks, class_names)
sns.heatmap(pd.DataFrame(matrix), annot=True, cmap="BuPu" ,fmt='g')
ax.xaxis.set_label_position("top")
plt.tight_layout()
plt.title('Confusion matrix', y=1.1)
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
plt.show()
```

The code creates a confusion matrix using'sklearn.metrics' to assess the classification performance of a logistic regression model. 'Seaborn' is used to visualize the matrix as a heat map, which aids in evaluating the model's ability to distinguish between phishing and legitimate URLs in the test data.

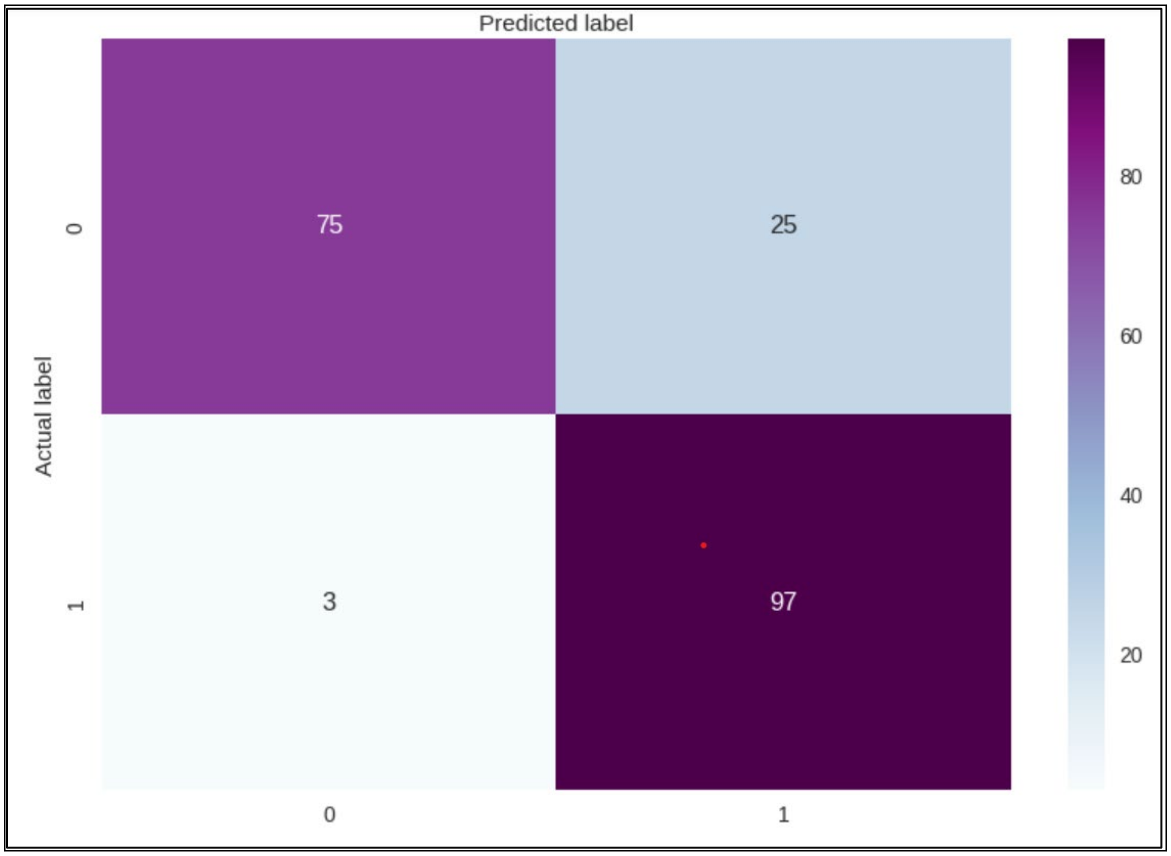


Figure 11 Output of confusion matrix

```
#Classification Report
print(classification_report(y_test, y_pred))
visualizer = ClassificationReport(logit, support=True, cmap="BuPu")
visualizer.fit(X_train, y_train)
visualizer.score(X_test, y_test)
g = visualizer.poof()
```

```
              precision    recall  f1-score   support

           0       0.80      0.87      0.83       100
           1       0.86      0.78      0.82       100

    accuracy                           0.82       200
   macro avg       0.83      0.82      0.82       200
weighted avg       0.83      0.82      0.82       200
```

A classification report for the performance of a logistic regression model is generated by the code snippet. The model uses'sklearn.metrics' to calculate precision, recall, and F1-score for phishing and legitimate URLs, and it also uses 'yellowbrick.classifier' to produce a visual report that helps evaluate the model's ability to distinguish between the two classes in test data.

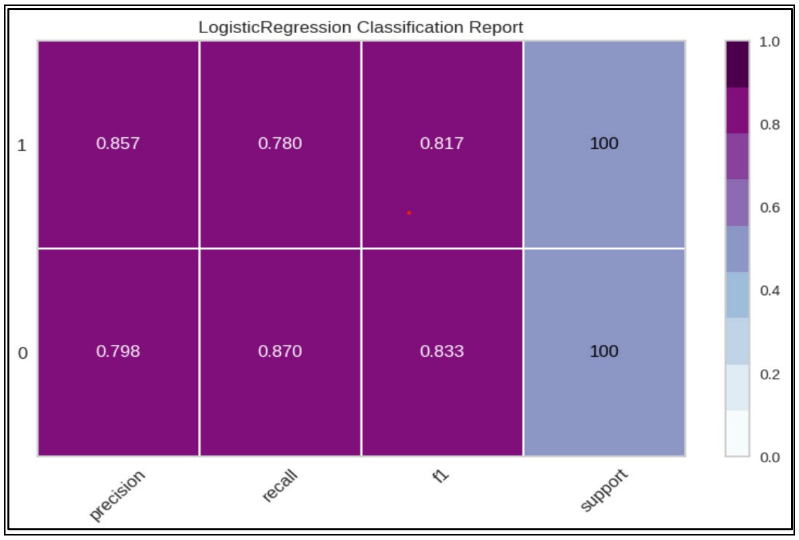
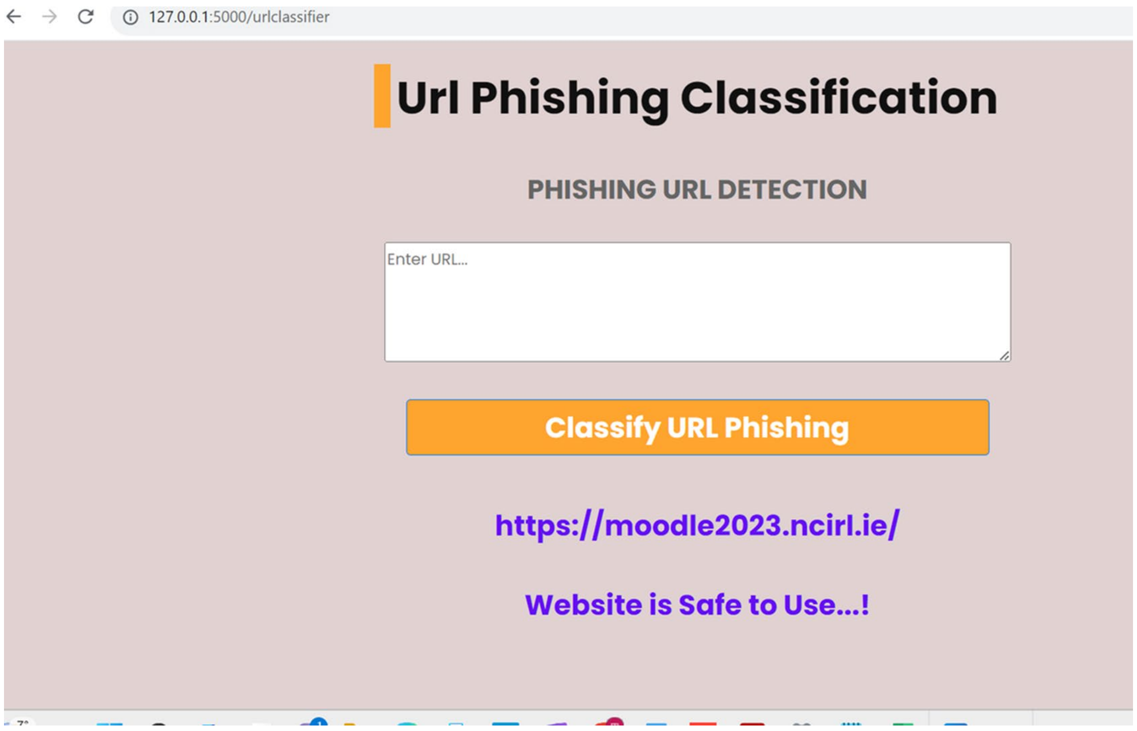


Figure 12 Classification Report of logistic regression

7. Visualization Techniques

Focus on confusion matrices and classification reports for model assessment. Utilizes Matplotlib and Seaborn for heatmap visualizations. Detailed analysis of true positives, negatives, and error distribution.

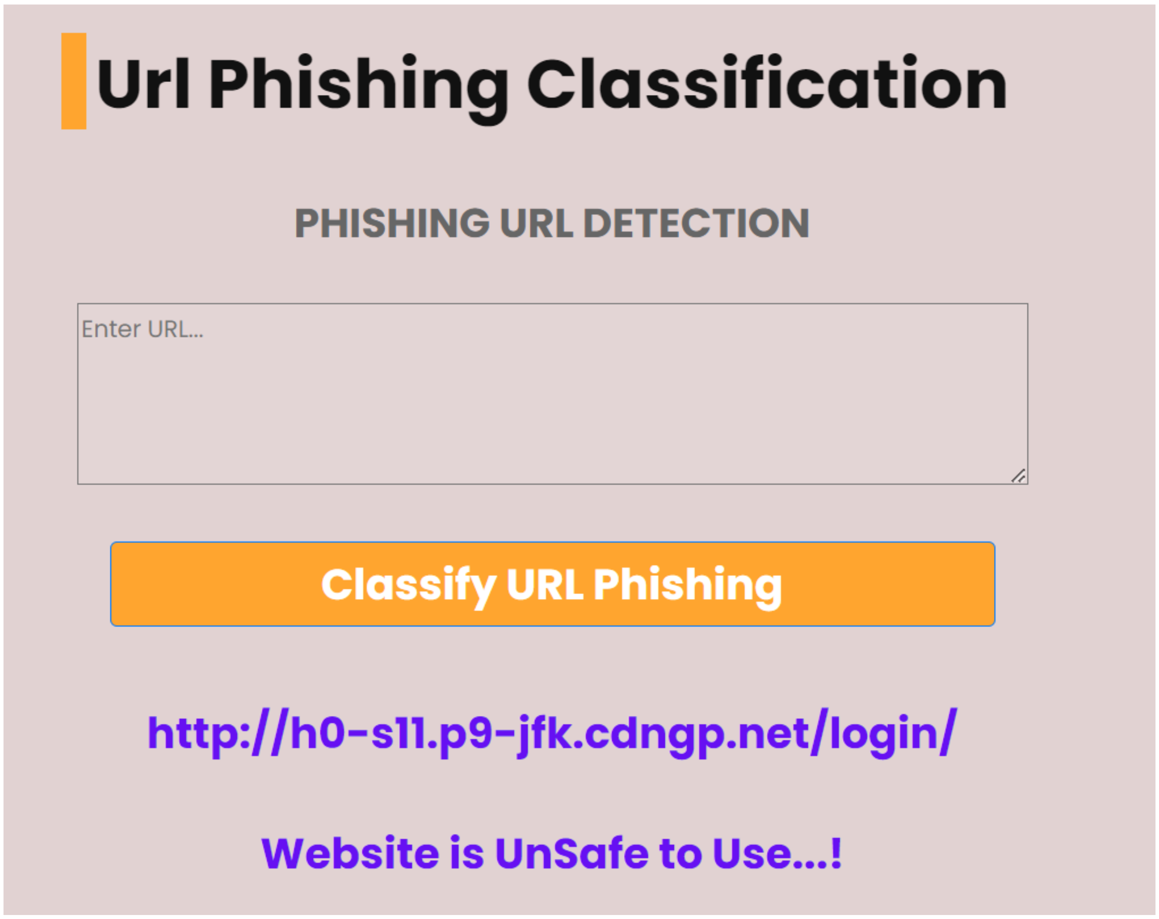
After running the python code and the successful training of the model, Now for the detection phase, whenever a website is visited, the URL of that website is transmitted to the feature extractor. Feature extractor extracts the required features of this currently visited website's URL. These extracted features are then transmitted to the classifier. Based on the knowledge gained by the classifier from its previous training, it decides whether the website is legitimate or not. It then displays a pop-up to the user based on its results and then a mail notification is sent to the user with the malicious URL link.



https://moodle2023.ncirl.ie/

Website is Safe to Use...!

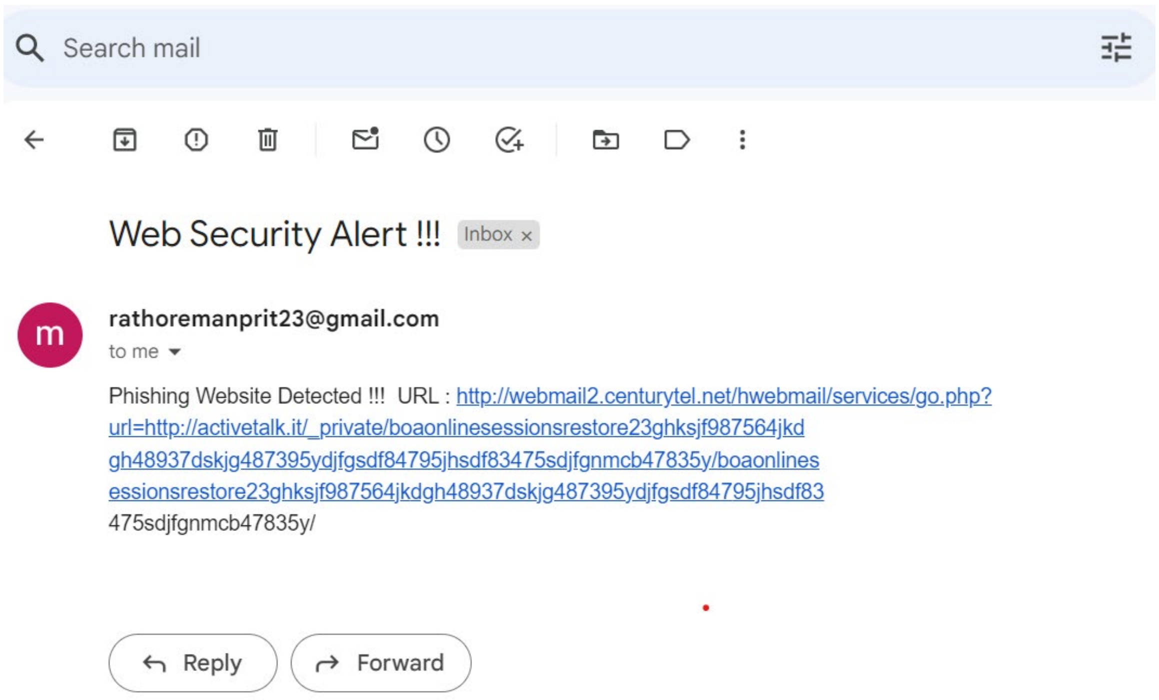
Figure 13 Output of the website detection



http://h0-s11.p9-jfk.cdngp.net/login/

Website is UnSafe to Use...!

Figure 14 Output when the website is unsafe



rathoremanprit23@gmail.com
to me

Phishing Website Detected !!! URL : http://webmail2.centurytel.net/hwebmail/services/go.php?url=http://activetalk.it/_private/boaonlinesessionsrestore23ghksjf987564jkdgh48937dskjg487395ydjfgsdf84795jhsdf83475sdjfgnmcb47835y/boaonlinesessionsrestore23ghksjf987564jkdgh48937dskjg487395ydjfgsdf84795jhsdf83475sdjfgnmcb47835y/

Figure 15 Output for the mail notification

References:

Introduction to NumPy. (n.d.). Retrieved December 14, 2023, from https://www.w3schools.com/python/numpy/numpy_intro.asp

Libraries in Python - GeeksforGeeks. (n.d.). Retrieved December 14, 2023, from https://www.geeksforgeeks.org/libraries-in-python/