Phishing Detection using Machine Learning algorithms

A Mini Project Report Submitted
In partial fulfillment of the requirement for the award of the degree of

Bachelor of Technology
In
Computer Science and Engineering -Artificial Intelligence and Machine Learning

by

Gadhari Kenneth	**-**	**20N31A6615**
Pasagada Deekshitha	**-**	**20N31A6646**

Under the Guidance of

Dr. P. Harikrishna
Assoc. Professor
Department of Computational Intelligence
MRCET

DEPARTMENT OF COMPUTATIONAL INTELLIGENCE
MALLA REDDY COLLEGE OF ENGINEERING AND TECHNOLOGY
(Affiliated to JNTU, Hyderabad)
ACCREDITED by AICTE-NBA
Maisammaguda, Dhulapally post, Secunderabad-500014.

2020-2024

DECLARATION

I hereby declare that the project entitled **"Phishing Detection using Machine Learning algorithms"** submitted to **Malla Reddy College of Engineering and Technology,** affiliated t**o** Jawaharlal Nehru Technological University Hyderabad (JNTUH) for the award of the degree of **Bachelor of Technology** in **Computer Science and Engineering- Artificial Intelligence and Machine Learning** is a result of original research work done by me.

It is further declared that the project report or any part thereof has not been previously submitted to any University or Institute for the award of degree or diploma.

G. Kenneth (20N31A6615)

P. Deekshitha (20N31A6646)

MALLA REDDY COLLEGE OF ENGINEERING & TECHNOLOGY
(AUTONOMOUS INSTITUTION - UGC, GOVT. OF INDIA)
Affiliated to JNTUH; Approved by AICTE, NBA-Tier 1 & NAAC with A-GRADE | ISO 9001:2015

CERTIFICATE

This is to certify that this is the bonafide record of the project titled **"Phishing Detection using Machine Learning algorithms"** submitted by **G. Kenneth (20N31A6615), P. Deekshitha (20N31A6646)** of B. Tech in the partial fulfillment of the requirements for the degree of **Bachelor of Technology** in **Computer Science and Engineering- Artificial Intelligence and Machine Learning**, Dept. of CI during the year 2023-2024. The results embodied in this project report have not been submitted to any other university or institute for the award of any degree or diploma.

Dr. P. Harikrishna
Assoc. Professor
INTERNAL GUIDE

Dr. D. Sujatha
HEAD OF THE DEPARTMENT

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

We feel honored and privileged to place our warm salutation to our college Malla Reddy College of Engineering and technology (UGC-Autonomous), our Director ***Dr. VSK Reddy*** who gave us the opportunity to have experience in engineering and profound technical knowledge.

We are indebted to our Principal ***Dr. S. Srinivasa Rao*** for providing us with facilities to do our project and his constant encouragement and moral support which motivated us to move forward with the project.

We would like to express our gratitude to our Head of the Department ***Dr. D. Sujatha*** for encouraging us in every aspect of our system development and helping us realize our full potential.

We would like to thank our application development guide as well as our internal guide

Dr. P. Harikrishna, for his structured guidance and never-ending encouragement. We are extremely grateful for valuable suggestions and unflinching co-operation throughout application development work.

We would also like to thank all supporting staff of department of CI and all other departments who have been helpful directly or indirectly in making our application development a success.

We would like to thank our parents and friends who have helped us with their valuable suggestions and support has been very helpful in various phases of the completion of the application development.

By

G. Kenneth (20N31A6615)
P. Deekshitha (20N31A6646)

ABSTRACT

In the world of today, the technology development led to the increase in number of scams. One of these includes “Phishing”. This is defined as a cybercrime when a target or targets are approached through email, phone call, or text message by someone posing as a reputable organization in order to trick people into disclosing sensitive information. This activity has grown more dangerous in the recent times, prompting various studies to identify the categorization method that best detects these attacks. Numerous research has compared only particular datasets and methodologies. As a result, if classification techniques done entirely on unique datasets and techniques, they cannot be generalized. Therefore, using a portion of the dataset's methods, we assessed how well classification approaches performed on dynamic data. In this project, we used certain schemes and the thirteen most recent categorization techniques (algorithms) were compared and assessed against 10 performance metrics. The findings of the project show the top and bottom performance levels attained by a categorization technique. This project is based on the domain of Machine learning.

TABLE OF CONTENTS

S. No. Topic Page No.

CHAPTER 1
INTRODUCTION

1.1 Purpose:

The main purpose of this document is to present the requirements of the project “Phishing detection using Machine Learning algorithms”. This study's goal is to evaluate the effectiveness of several phishing categorization methods using a range of different data sources and approaches. This study compares these strategies' efficiency in correctly recognizing phishing attempts by completing a thorough review. In order to identify the most effective strategies for preventing phishing threats in a variety of scenarios and datasets, the study aims to offer insights into which techniques function best under diverse conditions.

1.2 Background of project:

Phishing is a type of cybercrime when criminals attempt to gain sensitive or private information from consumers by establishing a fake website that seems identical to a legitimate one. Multiple tactics, including link manipulation, filter evasion, website forgery, covert redirect, and social engineering, are used in phishing assaults.

The escalating threat of phishing attacks necessitates effective classification techniques. In addition to this, there is a pressing need to comprehensively evaluate the performance of phishing classification techniques across diverse scenarios, enabling the identification of techniques that yield accurate and consistent results across various data sources and classification schemes.

The existing works based on phishing detection using classification methods are mostly based on heuristic features where feature selection can be extracted from the attributes such as Uniform Resource Locator, Source Code, Session, type of security involve, protocol used, type of website. Limited number of machine learning algorithms have been used to test their model and the metrics used are only regression tasks. The proposed system project assess the classification methods on dataset through schemes. The ten performance measures are compared and evaluated between the thirteen most recent classification techniques. The proposed system helps users in providing all-inclusive views of model performance.

1.3 Scope of project:

The goals of the project were accomplished based on the finding that the use of a subset scheme can affect how effectively classification techniques perform on various datasets. The users come to know the ability of the classification methods which perform using a subset scheme applied on balanced and balanced datasets. The scheme is applied on the datasets in the sequence of phishing: legitimate and legitimate: phishing. Users can have a tendency to significantly improve and reduce performance. The results of this study demonstrate that performance during the categorization process is severely impacted by unused data. In future, this subset scheme needs to be applied to confirmed cases to compare its performances. In addition, certain recommendations were proposed for developing research on improving phishing classification techniques.

1.4 Project Features:

The system features are as follows:

- Data Collection: It is the process of gathering and analysing data on pertinent variables in a planned, methodical manner in order to address certain research questions, test hypotheses, and evaluate outcomes. Both qualitative and quantitative data is collected in this product.
- Data preprocessing: Data preprocessing is the process of cleaning, transforming, and organizing raw data before it is analysed. It involves removing irrelevant data, correcting errors, handling missing values, and transforming data into a suitable format for analysis.
- Training and Testing: In order to actually train and test a model, we'll need to perform a train test split, where we will split data, we have into two groups - train data and test data.
- Modelling: In this step, machine learning techniques are used to create a prediction model. This involves choosing the best method, building the model using the data, and assessing how well it performs. This step is crucial because it entails creating a model that can precisely predict outcomes on fresh data, which is the core of the product.
- Predicting: Prediction is the final feature in this product. It is employed to discover a numerical result. The training dataset includes the inputs and matching numerical output values, much like in classification. Using the training dataset, the algorithm creates a model or prediction. When the new data is provided, the model should produce a numerical result. A continuous-valued function or an ordered value is predicted by the model.

CHAPTER 2
SYSTEM REQUIREMENTS

2.1 Hardware Requirements:

The hardware interfaces of this product consist of architecture, processing power, memory, secondary storage, display adapter, peripherals like CD-ROM drivers, keyboards, pointing devices, network devices, etc.

- Processor: i3 and above
- RAM: 8GB and above
- Hard Disk: 40 GB and above in local drive

2.2 Software requirements:

The software interfaces of this product consist of Platform, APIs and drivers, Web browser (Anaconda).

- Operating System: Windows Only

2.3 Existing System:

The existing system has been made using five machine learning algorithms such as Random forest, K-Nearest Neighbor, Decision Tree, Support Vector Machine, Logistic regression and these were implemented by are implemented by the Performance metrics like Root Mean Square Error (RMSE), R-Squared, Mean Absolute Error (MAE) and Mean Squared Error (MSE). In this system a new classification algorithm has been discovered using feature extraction method and the above algorithms.

2.3.1 Drawbacks of existing system:

- The present work employs interesting features collected from attributes like URL, Source Code, Session, etc., which may limit the approach's adaptability and generalizability.
- The existing work consists of limited machine learning algorithms and metrics which are associated with regression tasks.
- This work has less robustness because it does not deal with the diversity of datasets which are used for evaluation.

2.4 Proposed System:

The proposed system assesses the performance of different phishing classification techniques across diverse data sources and schemes. This system aims to compare the effectiveness and efficiency of these techniques in accurately identifying phishing attacks. A subset of schemes is applied on the datasets. Both balanced and unbalanced datasets are used in this proposed system. The datasets are MDP- 2018, UCI Phishing website, and Spam base. MDP 2018 is a balanced dataset, whereas the UCI Phishing website and Spam base datasets are unbalanced datasets. The subset schemes are implemented in the ratios of 90:10, 80:20, 70:30, and 60:40. The ten performance measures i.e., metrics such as accuracy, F-Measure, Precision, True Positive Rate (TPR), Receiver Operating Characteristic (ROC), False Positive Rate (FPR), Precision-Recall Curve (PRC), Matthews Correlation Coefficient (MCC), Balanced Detection Rate (BDR), and Geometric Mean (G-Mean) are being evaluated and compared using thirteen most recent classification techniques namely, Random forest, SVM, Logistic regression, MLP, C4.5, Bayesian Network, REP Tree, Naïve Bayes, P.A.R.T, ABET (AdaBoost.M1 and Extra trees), ROFET (Rotation Forest and Extra trees), BET (Bagging and Extra-trees) and LBET (Logit Boost and Extra trees). The proposed system helps users in providing a more holistic view of model performance.

CHAPTER 3
TECHNOLOGIES USED

3.1 Feature Extraction

Feature extraction refers to the process of transforming raw data into numerical features that can be processed while preserving the information in the original data set. It yields better results than applying machine learning directly to the raw data. Machine learning models require relevant features or attributes from the data to make predictions. For phishing detection, these features could include email headers, URLs, content, and metadata.

3.2 URL Analysis

Analyzing URLs in emails and websites is critical for identifying phishing attempts. Techniques like URL parsing, domain analysis, and blacklisting can be used to assess the legitimacy of links.

CHAPTER 4
SYSTEM DESIGN

4.1 System Architecture

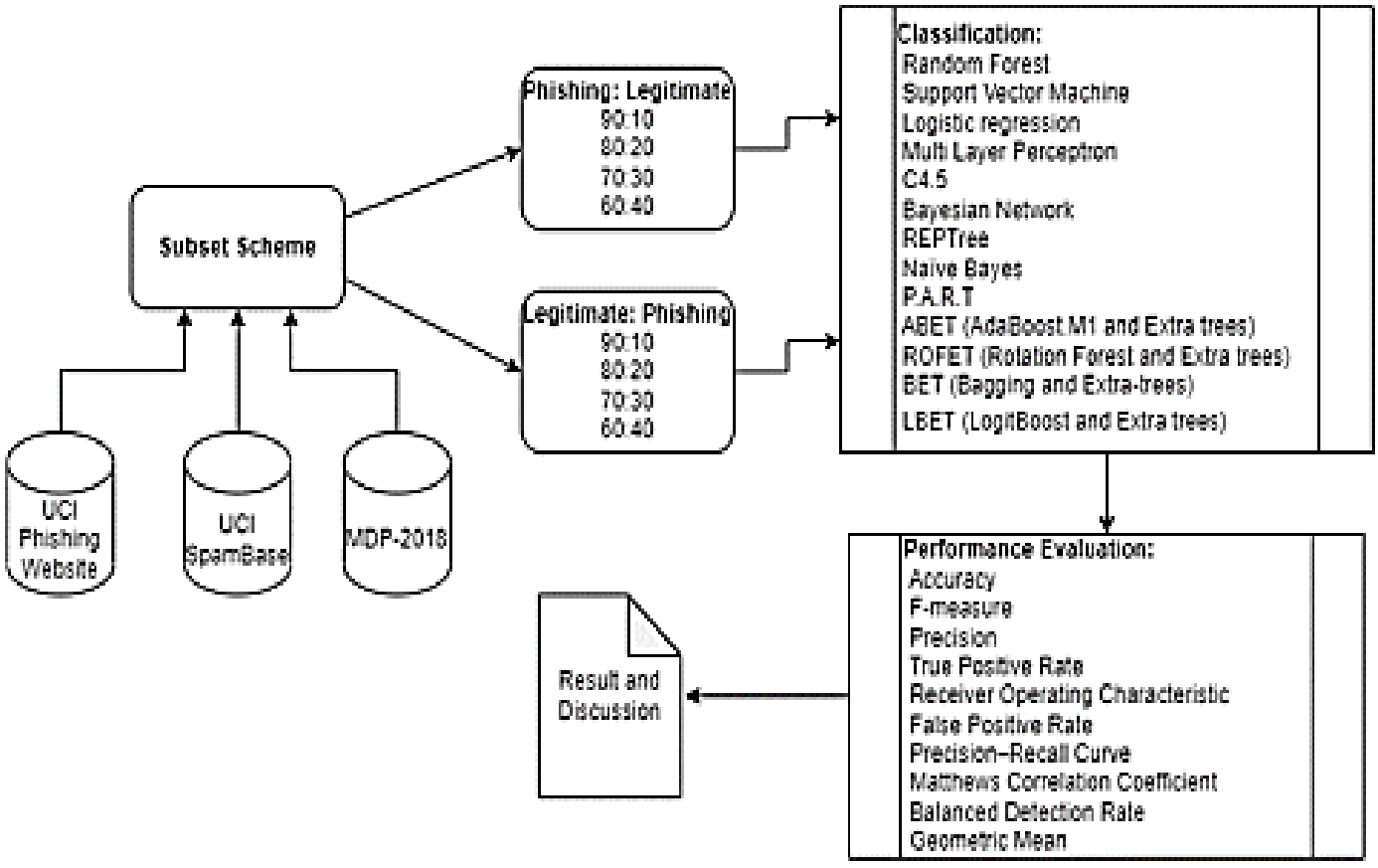


Fig 4.1: System architecture

4.2 Data Flow Diagram

DFD represents the data flow of a system or process. It also sheds light on each entity's inputs, outputs, and the process itself. There are no loops, decision rules, or control flows in DFD. A flowchart can describe specific operations depending on the type of data. It is a graphic tool that can be used to communicate with users, supervisors, and other staff members. It can be used to analyze both current and new systems.

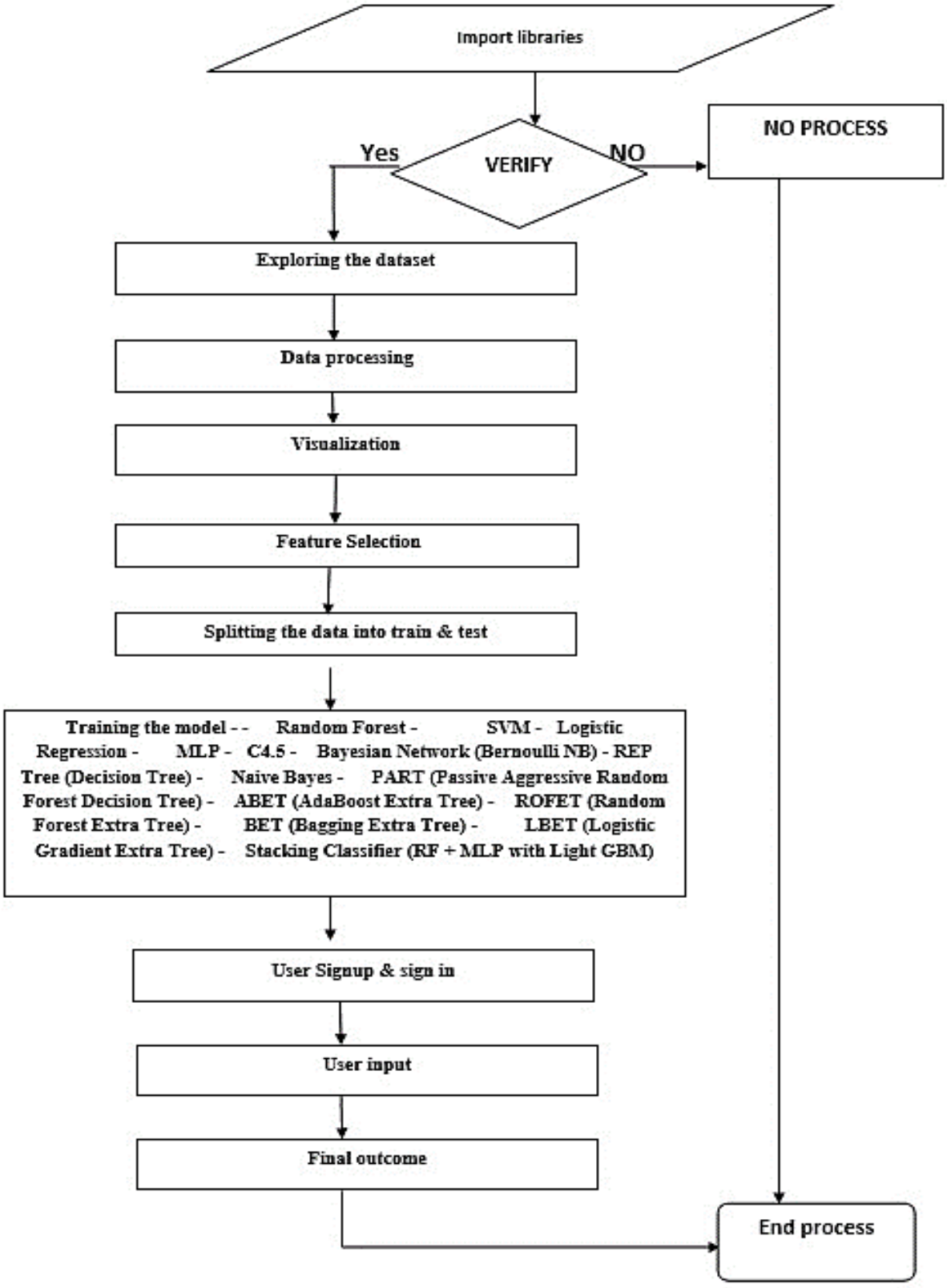


Fig 4.2: Data flow diagram

4.3 UML Diagrams

4.3.1 Use Case Diagram

Use Case during requirement elicitation and analysis to represent the functionality of the system. Use case describes a function by the system that yields a visible result for an actor. The identification of actors and use cases result in the definitions of the boundary of the system i.e., differentiating the tasks accomplished by the system and the tasks accomplished by its environment.

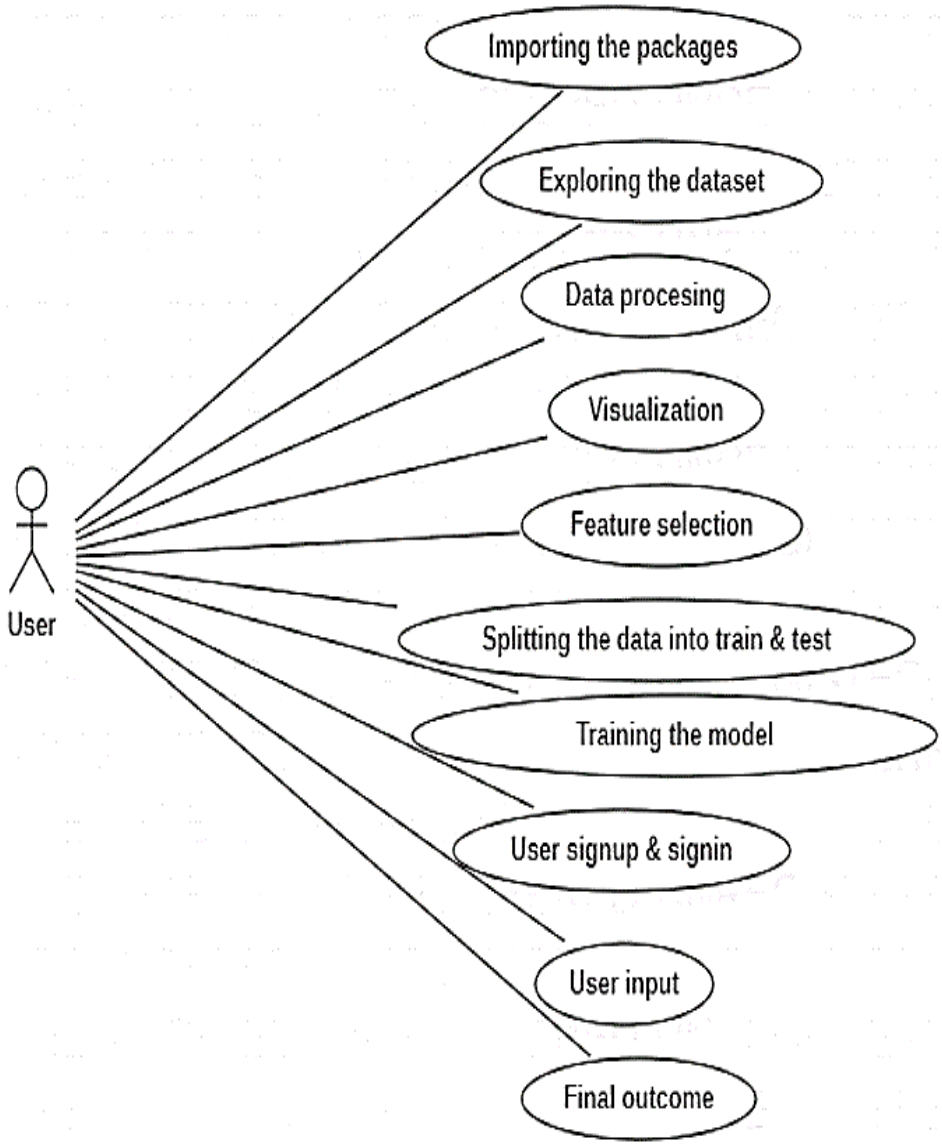


Fig 4.3.1: Use case diagram

4.3.2 Class Diagram

Class diagrams model class structure and contents using design elements such as classes, packages and objects. Class diagram describe the different perspective when designing a system-conceptual, specification and implementation. Classes are composed of three things: name, attributes, and operations. Class diagram also display relationships such as containment, inheritance, association etc. The association relationship is most common relationship in a class diagram. The association shows the relationship between instances of classes.

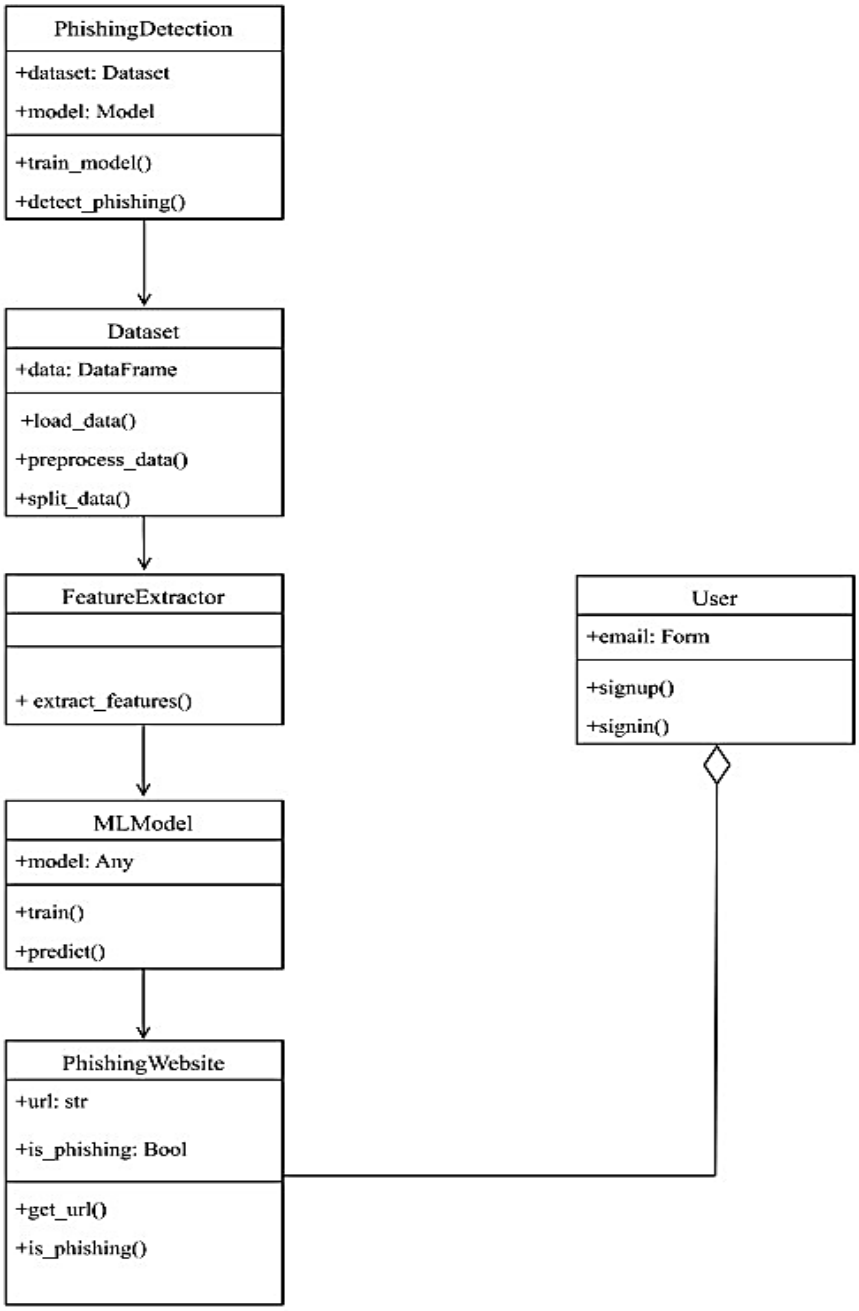


Fig 4.3.2: Class diagram

4.3.3 Sequence Diagram

Sequence diagram displays the time sequence of the objects participating in the interaction. This consists of the vertical dimension (time) and horizontal dimension (different objects).

Objects: An object can be thought of as an entity that exists at a specified time and has a definite value, as well as a holder of identity. A sequence diagram depicts item interactions in chronological order. It illustrates the scenario's objects and classes, as well as the sequence of messages sent between them in order to carry out the scenario's functionality. In the Logical View of the system under development, sequence diagrams are often related with use case realizations. Event diagrams and event scenarios are other names for sequence diagrams. A sequence diagram depicts multiple processes or things that exist simultaneously as parallel vertical lines (lifelines), and the messages passed between them as horizontal arrows, in the order in which they occur. This enables for the graphical specification of simple runtime scenarios.

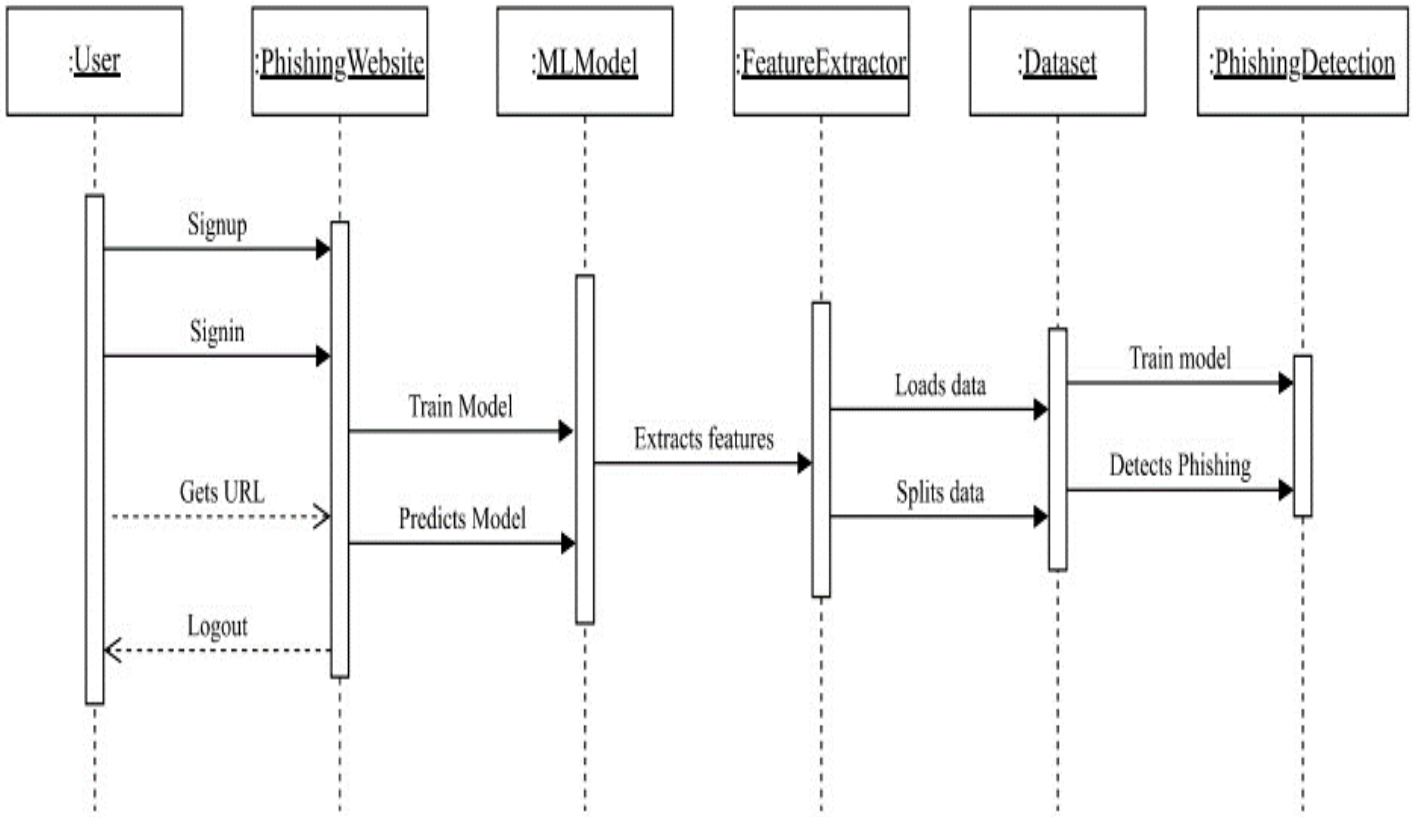


Fig 4.3.3: Sequence diagram

4.3.4 Activity Diagram

The process flows in the system are captured in the activity diagram. Similar to a state diagram, an activity diagram also consists of activities, actions, transitions, initial and final states, and guard conditions.

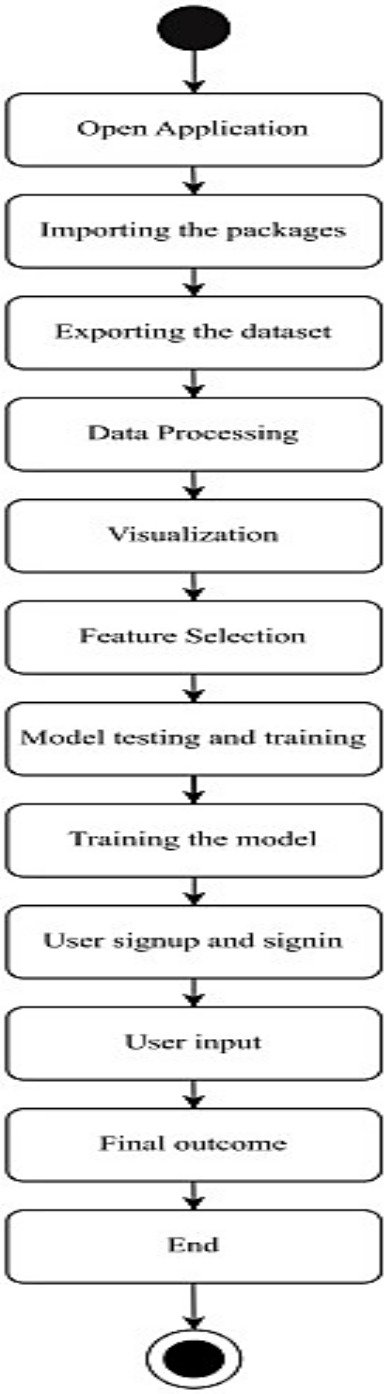


Fig 4.3.4: Activity diagram

CHAPTER 5
IMPLEMENTATION

5.1: Source Code

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

import pandas as pd
import numpy as np
import time
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.graph_objects as go
import plotly.express as px
from plotly.subplots import make_subplots

from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.decomposition import PCA

from sklearn.model_selection import KFold
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
from sklearn.neural_network import MLPClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA

from sklearn.metrics import recall_score
from sklearn.metrics import precision_score
from sklearn.metrics import f1_score
from sklearn.metrics import accuracy_score
from sklearn.model_selection import StratifiedKFold
from scipy.stats import friedmanchisquare
from scipy import stats
from sklearn.metrics import confusion_matrix

import keras
from keras.models import Sequential
from keras.layers import Dense
```

```
# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

df = pd.read_csv('/kaggle/input/website-phishing-data-set/Website Phishing.csv')
# Replacing the phishy class by the value 2
# I decided to use MLP so it would be better if
# the classes were all non-negative values

df['Result'] = df['Result'].astype(str).replace('-1','2').astype(np.int64)
# Visualizing the data
df.head()
# Visualizing the data
df.tail()
#  Calculating if there are null or na values in the dataset
print('Verifying null and na data')
print()
print(df.isna().any())

print()
print(df.isnull().any())

# Obtaining some additional information about the dataset
df.info()
def process_data_class_distribution(df):
    class_names = ['Legitimate','Suspicious', 'Phishy']

    class_samples = [len(df[df['Result'] == 1]), len(df[df['Result'] == 0]), len(df[df['Result'] == 2])]

    data = {
      'Class': class_names,
      'Samples': class_samples,
    }

    df_class = pd.DataFrame(data)

    df_class.sort_values(by='Samples', ascending=True, inplace=True)

    return df_class

def fig_class_distribution(df):
    fig = go.Figure()
```

```
    for idx,classe in enumerate(list(df.Class.unique())):

        if idx == 0: # the class with less samples will be highlighted
            color = 'rgb(90, 90, 200)'

        else:
            color = 'rgb(120, 120, 120)'

        fig.add_trace(go.Bar(y=[classe], x=df[df['Class'] == classe]['Samples'], marker_color=color, name=classe,
orientation='h'))

    return fig

def update_layout_bar_chart(fig):
    fig.update_layout(
        title= 'Class Distribution',
        font=dict(
            family="Arial",
            size=16),
        plot_bgcolor="#FFFFFF",
        margin=dict(l=20, r=20, b=20, t=50),
        width=600,
        height=400,

        hoverlabel=dict(
            font_color= 'rgb(120, 120, 120)',
            bgcolor="white",
            font_size=16,
            font_family="Arial"
        ),
        xaxis=dict(

            showline=True,
            showgrid=False,
            showticklabels=True,
            linecolor='rgb(204, 204, 204)',
            linewidth=2,
            ticks='outside',
            tickfont=dict(
                family='Arial',
                size=16,
                color='rgb(82, 82, 82)',),
        ),

        yaxis=dict(

            showline=True,
```

```
            showgrid=False,
            gridcolor='rgb(204, 204, 204)',
            gridwidth=2,
            showticklabels=True,
            linecolor='rgb(204, 204, 204)',
            linewidth=2,
            ticks=None,
            tickfont=dict(
                family='Arial',
                size=16,
                color='rgb(82, 82, 82)',
            ),

    ))
    fig.update_layout(showlegend=False)
    fig.update_layout(yaxis_title='Class')
    fig.update_layout(xaxis_title ='Number of Samples')

def plot_class_distribution(df):

    df_class = process_data_class_distribution(df)
    fig = fig_class_distribution(df_class)

    update_layout_bar_chart(fig)

    fig.show()

plot_class_distribution(df)
features = ['SFH', 'popUpWidnow', 'SSLfinal_State', 'Request_URL', 'URL_of_Anchor',
       'web_traffic', 'URL_Length', 'age_of_domain', 'having_IP_Address',]
df_corr = df.loc[:, features].corr()

sns.heatmap(df_corr)
plt.show()
X = df.drop(['Result' ], axis=1)
y = df['Result']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=42)

model = LogisticRegression()
model.fit(X_train, y_train.values)

importances = pd.DataFrame(data={
    'Attribute': X_train.columns,
    'Importance': model.coef_[0]
})
importances = importances.sort_values(by='Importance', ascending=False)
phishy_importances = pd.DataFrame(data={
```

```
    'Attribute': X_train.columns,
    'Importance': model.coef_[2]
})
phishy_importances = phishy_importances.sort_values(by='Importance', ascending=False)
phishy_importances
legit_importances = pd.DataFrame(data={
    'Attribute': X_train.columns,
    'Importance': model.coef_[1]
})
legit_importances = legit_importances.sort_values(by='Importance', ascending=False)
legit_importances
x = df.loc[:, features].values
pca_model = PCA(n_components=2)
principalComponents_breast = pca_model.fit_transform(x)

principal_df = pd.DataFrame(data = principalComponents_breast
            , columns = ['Principal component 1', 'Principal component 2'])
df_pca = pd.concat([df, principal_df], axis=1)
fig = go.Figure()

fig.add_trace(go.Scatter(x=df_pca[df_pca['Result'] == 0]['Principal component 1'], y=df_pca[df_pca['Result']
== 0]['Principal component 2'],name='Suspicious', mode='markers',  marker_color='#ffdd00'))
fig.add_trace(go.Scatter(x=df_pca[df_pca['Result'] == 2]['Principal component 1'], y=df_pca[df_pca['Result']
== 2]['Principal component 2'],name='Phishy', mode='markers', marker_color='#d30d0d'))
fig.add_trace(go.Scatter(x=df_pca[df_pca['Result'] == 1]['Principal component 1'], y=df_pca[df_pca['Result']
== 1]['Principal component 2'],name='Legitimate', mode='markers', marker_color='#55a630'))

fig.update_layout(
      title= 'Samples per class by Principal Component Analysis',
      font=dict(
          family="Arial",
          size=16),
      plot_bgcolor="#FFFFFF",
      margin=dict(l=20, r=20, b=20, t=50),
      width=600,
      height=400,

      hoverlabel=dict(
          font_color= 'rgb(120, 120, 120)',
          bgcolor="white",
          font_size=16,
          font_family="Arial"
      ),
      xaxis=dict(

          showline=True,
          showgrid=False,
```

```
        showticklabels=True,
        linecolor='rgb(204, 204, 204)',
        linewidth=2,
        ticks='outside',
        tickfont=dict(
            family='Arial',
            size=16,
            color='rgb(82, 82, 82)',),
    ),

    yaxis=dict(

        showline=True,
        showgrid=False,
        gridcolor='rgb(204, 204, 204)',
        gridwidth=2,
        showticklabels=True,
        linecolor='rgb(204, 204, 204)',
        linewidth=2,
        ticks=None,
        tickfont=dict(
            family='Arial',
            size=16,
            color='rgb(82, 82, 82)',
        ),

 ))
fig.update_layout(showlegend=True)
fig.update_layout(yaxis_title='Principal Component 2')
fig.update_layout(xaxis_title ='Principal Component 1')
fig.show()


fig = make_subplots(rows=1, cols=3,subplot_titles=( "Phishy - PCA","Suspicious - PCA ", "Legitimate -
PCA"), start_cell="bottom-left")

fig.add_trace(go.Scatter(x=df_pca[df_pca['Result'] == 2]['Principal component 1'], y=df_pca[df_pca['Result']
== 2]['Principal component 2'],name='Phishy', mode='markers', marker_color='#d30d0d'), row=1, col=1)
fig.add_trace(go.Scatter(x=df_pca[df_pca['Result'] == 0]['Principal component 1'], y=df_pca[df_pca['Result']
== 0]['Principal component 2'],name='Suspicious', mode='markers',  marker_color='#ffdd00'), row=1, col=2)
fig.add_trace(go.Scatter(x=df_pca[df_pca['Result'] == 1]['Principal component 1'], y=df_pca[df_pca['Result']
== 1]['Principal component 2'],name='Legitimate', mode='markers', marker_color='#55a630'), row=1, col=3)

fig.update_xaxes(range=[-3, 3])
fig.update_yaxes(range=[-3, 3])
fig.update_layout(height=400, width=1050, template='plotly_white')
```

```
fig.show()
# function to plot the class distribution by feature

def autolabel(rects, ax):
    for rect in rects:
        height = rect.get_height()
        ax.text(rect.get_x() + rect.get_width()/2., 1.05*height,
                '%.f' % float(height),
        ha='center', va='bottom')

def plot_class_distribution(feature, color, data, labels):

    class_info = data[feature].value_counts().sort_index()

    x = labels
    x_pos = [i for i, _ in enumerate(x)]

    y = class_info.values

    fig, ax = plt.subplots()
    rects1 = ax.bar(x_pos, y, color=color)

    autolabel(rects1, ax)

    plt.ylabel("Number of Examples")
    plt.title(feature + " examples distribution\n")
    plt.xticks(x_pos, x)

# Distribution of the column values according to the description from the paper

sfh_labels = ['Empty SFH', 'SFH different domain', 'Valid SFH']
plot_class_distribution('SFH', 'silver', df, sfh_labels)

pop_labels = ['Rightclick disabled', 'Rightclicl with alert', 'No pop-up']
plot_class_distribution('popUpWidnow', 'gold', df, pop_labels)

ssl_labels = ['Nor HTTP nor trusted', 'HTTP and nottrusted', 'HTTP and trusted']
plot_class_distribution('SSLfinal_State', 'silver', df, ssl_labels)

request_labels = ['req_URL > 61%',  '22 <= req_URL <= 61%', 'req_URL < 22%']
plot_class_distribution('Request_URL', 'gold', df, request_labels)

anchor_labels = [ 'Acr_URL>67%',' 31%<=Acr_URL<=67%', 'Acr_URL<31%']
plot_class_distribution('URL_of_Anchor', 'silver',df, anchor_labels)

web_labels = ['wtraffic>150K', 'wtraffic<=150K', 'wtraffic<150K']
plot_class_distribution('web_traffic', 'gold', df, web_labels)
```

```
url_labels = ['len > 75', '54 <= len <= 75', 'len < 54']
plot_class_distribution('URL_Length', 'silver', df, url_labels)

age_labels = ['age < 1 year', 'age > 1 year']
plot_class_distribution('age_of_domain', 'lightblue', df, age_labels)

ip_labels = ['No IPAdress URL','URL IPaddress']
plot_class_distribution('having_IP_Address', 'lightblue', df, ip_labels)

from sklearn.ensemble import AdaBoostClassifier
import tensorflow
from tensorflow.keras.optimizers import Adam
from keras.utils import np_utils
def initialize_classifiers():

    adab = AdaBoostClassifier(n_estimators=100)

    # Support Vector Machine - função de kernel RBF
    svmRBF = SVC(kernel='rbf')

    # Random Forest
    randomForest = RandomForestClassifier()

    # Logistic Regression
    logisticRegression = LogisticRegression()

    # MLP
    modelo = Sequential()
    modelo.add(Dense(units=64, activation='relu',kernel_initializer='random_uniform', input_dim=5))
    modelo.add(Dense(units=32, activation='relu',kernel_initializer='random_uniform'))
    modelo.add(Dense(units=16, activation='relu',kernel_initializer='random_uniform'))
    modelo.add(Dense(units=2, activation='sigmoid'))

    optimizer = tensorflow.keras.optimizers.Adam(lr = 0.001, decay =0.0001, clipvalue= 0.5)
    modelo.compile(optimizer=optimizer, loss='binary_crossentropy', metrics=['accuracy'])

    return [adab, svmRBF, randomForest, logisticRegression, modelo]

def train_and_evaluate(class_train, noclass_train, class_test, noclass_test, alg, dicio, epochs = 400, batch_size =
64, verb = False):
    if dicio['modelname'] == 'MLP':

        class_train_categorical = keras.utils.np_utils.to_categorical(class_train, num_classes=0)
        class_test_categorical = keras.utils.np_utils.to_categorical(class_test, num_classes=0)
        tempo_inicial = time.time()
        alg.fit(noclass_train, class_train_categorical,batch_size=batch_size, epochs=epochs, verbose= verb)
```

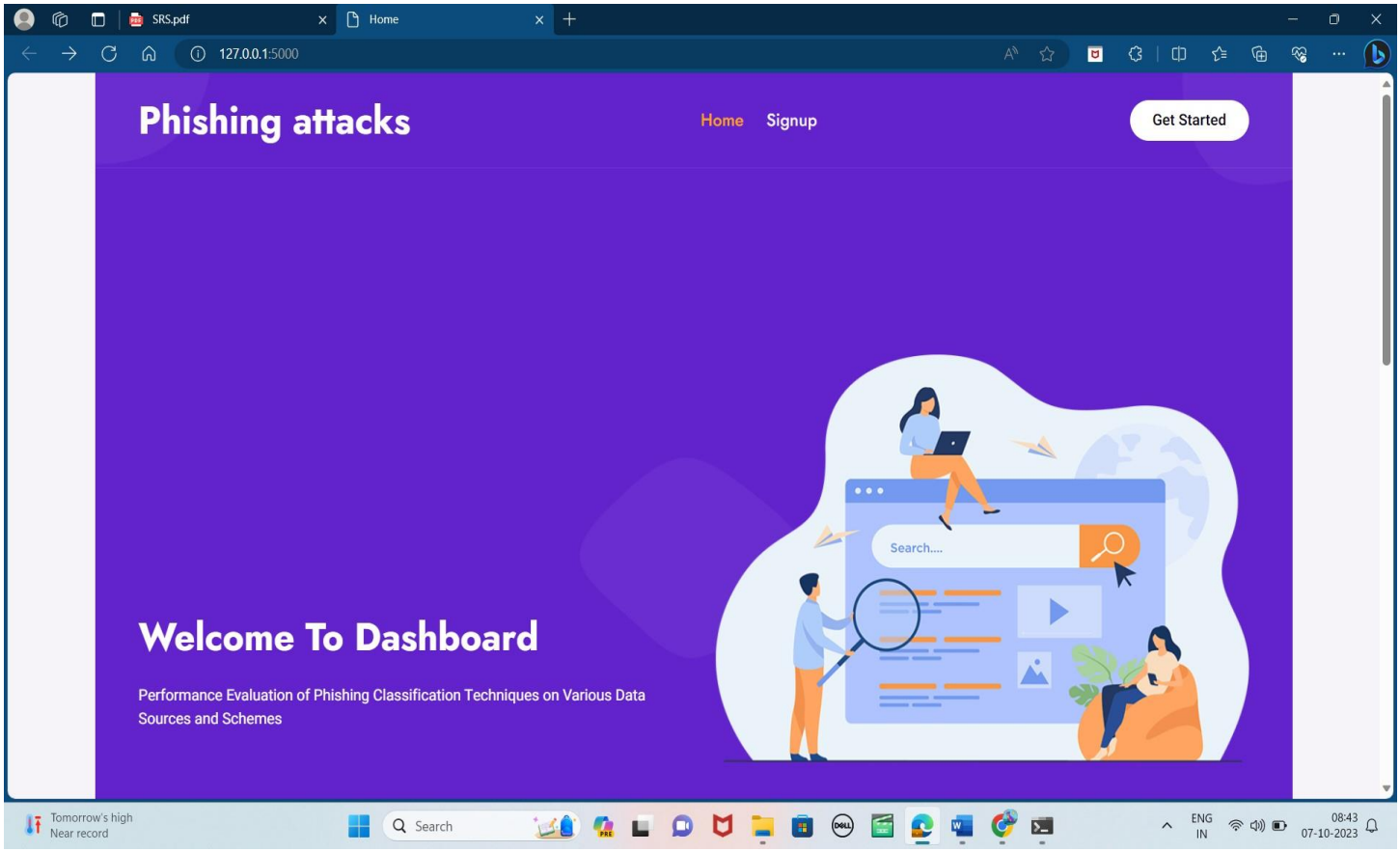
```
            tempo_fim = time.time()
            predicted = [np.argmax(pred) for pred in alg.predict(noclass_test)]
        else:
            tempo_inicial = time.time()
            alg.fit(noclass_train, class_train)
            tempo_fim = time.time()

            predicted = alg.predict(noclass_test)

        dicio['acc'].append(accuracy_score(class_test, predicted))
        dicio['fscore'].append(f1_score(class_test, predicted, average='macro'))
        dicio['precision'].append(precision_score(class_test, predicted, average='macro'))
        dicio['recall'].append(recall_score(class_test, predicted, average='macro'))
        dicio['tempo'].append(tempo_fim - tempo_inicial)
        dicio['cm'].append(confusion_matrix(class_test,predicted))

# Creating dictionaies to store the metric data to each one of the classifiers

def create_dictios():
    adab_dc =  {'modelname':'ADA-B','acc': [], 'fscore': [], 'precision': [], 'recall': [] , 'tempo':[], 'cm': []}
    svmRBF_dc = {'modelname':'SVMR','acc': [], 'fscore': [], 'precision': [], 'recall': [] , 'tempo':[], 'cm': []}
    randomForest_dc = {'modelname':'RF','acc': [], 'fscore': [], 'precision': [], 'recall': [], 'tempo':[], 'cm': []}
    logreg_dc = {'modelname':'LR','acc': [], 'fscore': [], 'precision': [], 'recall': [], 'tempo':[], 'cm': []}
    mlp_dc = {'modelname':'MLP','acc': [], 'fscore': [], 'precision': [], 'recall': [], 'tempo':[], 'cm': []}

    return [adab_dc, svmRBF_dc, randomForest_dc, logreg_dc, mlp_dc]

######## Data visualization functions ##########

# Creating boxplot with the data from a metric

def create_boxplot(df, metric_name):
    df.boxplot(column= list(df.columns) , figsize=(12, 8))
    plt.xlabel('Algorithms')
    plt.ylabel(metric_name)
    plt.show()

    print()
    print()
    print()

# Creating the statistic test to a pair of classifiers

def create_statistic_test(measure, df):

    two_bests = list(df.mean().sort_values(ascending = False).index[:2])
    best = df[two_bests[0]].values
```

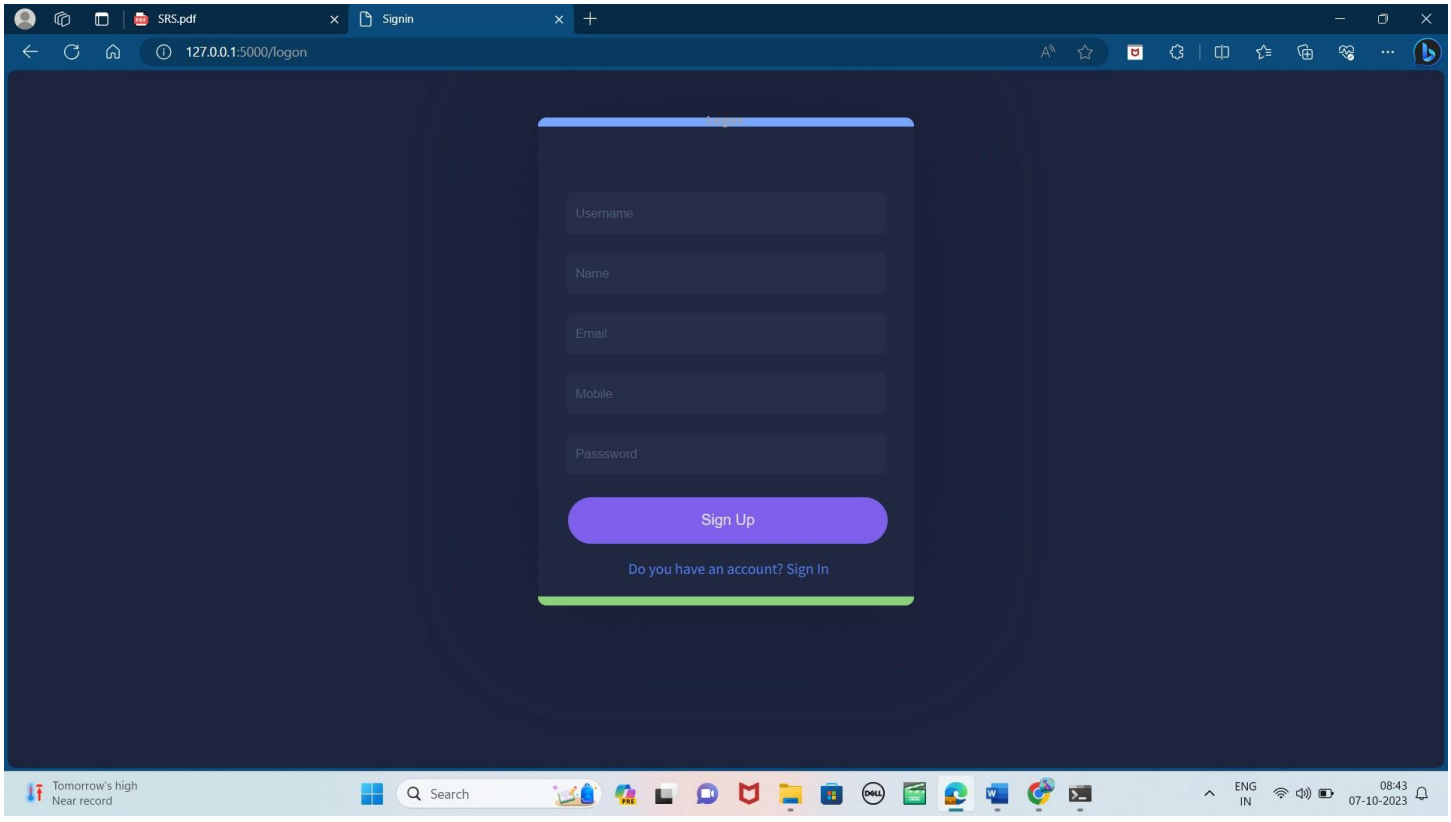
```
    secondbest = df[two_bests[1]].values

    pvalue = stats.ttest_ind(best, secondbest)
    if pvalue[1] < 0.05:
        print('The algorithms '+ two_bests[0] + ' and ' + two_bests[1] + ' are statistically different using the metric '
+ measure)
        print('P-value: ' + str(pvalue[1]))
    else:
        print('The algorithms '+ two_bests[0] + ' and ' + two_bests[1] + ' are  NOT statistically different using the
metric '  + measure)
        print('P-value: ' + str(pvalue[1]))

# Removing data from the suspicious class
df_binary = df[df['Result'] != 0]

df_binary['Result'] = [0 if x == 2 else 1 for x in df_binary['Result']]
# Splitting the data into the features and classes
important_features = list(phishy_importances['Attribute'].values[:5])

df_noclass = df_binary.loc[:, important_features]

df_class = df_binary.loc[:, 'Result']
# Classification with k-fold cross validation

# Initializing the classifiers
listmodels = initialize_classifiers()

# Creating the dictionaries that will store the classifiers' results
listofdicts = create_dictios()

# Initializing the K-fold
kfold = StratifiedKFold(10, shuffle=True, random_state=1)

c = kfold.split(df_noclass, df_class)

for train_index, test_index in c:

    noclass_train, noclass_test =np.array(df_noclass.iloc[train_index]) ,np.array(df_noclass.iloc[test_index])
    class_train, class_test = np.array(df_class.iloc[train_index]), np.array(df_class.iloc[test_index])

    train_and_evaluate(class_train, noclass_train, class_test, noclass_test, listmodels[0], listofdicts[0])
    train_and_evaluate(class_train, noclass_train, class_test, noclass_test, listmodels[1], listofdicts[1])
    train_and_evaluate(class_train, noclass_train, class_test, noclass_test, listmodels[2], listofdicts[2])
    train_and_evaluate(class_train, noclass_train, class_test, noclass_test, listmodels[3], listofdicts[3])
    train_and_evaluate(class_train, noclass_train, class_test, noclass_test, listmodels[4] , listofdicts[4], epochs =
400, batch_size = 64, verb = True)
```

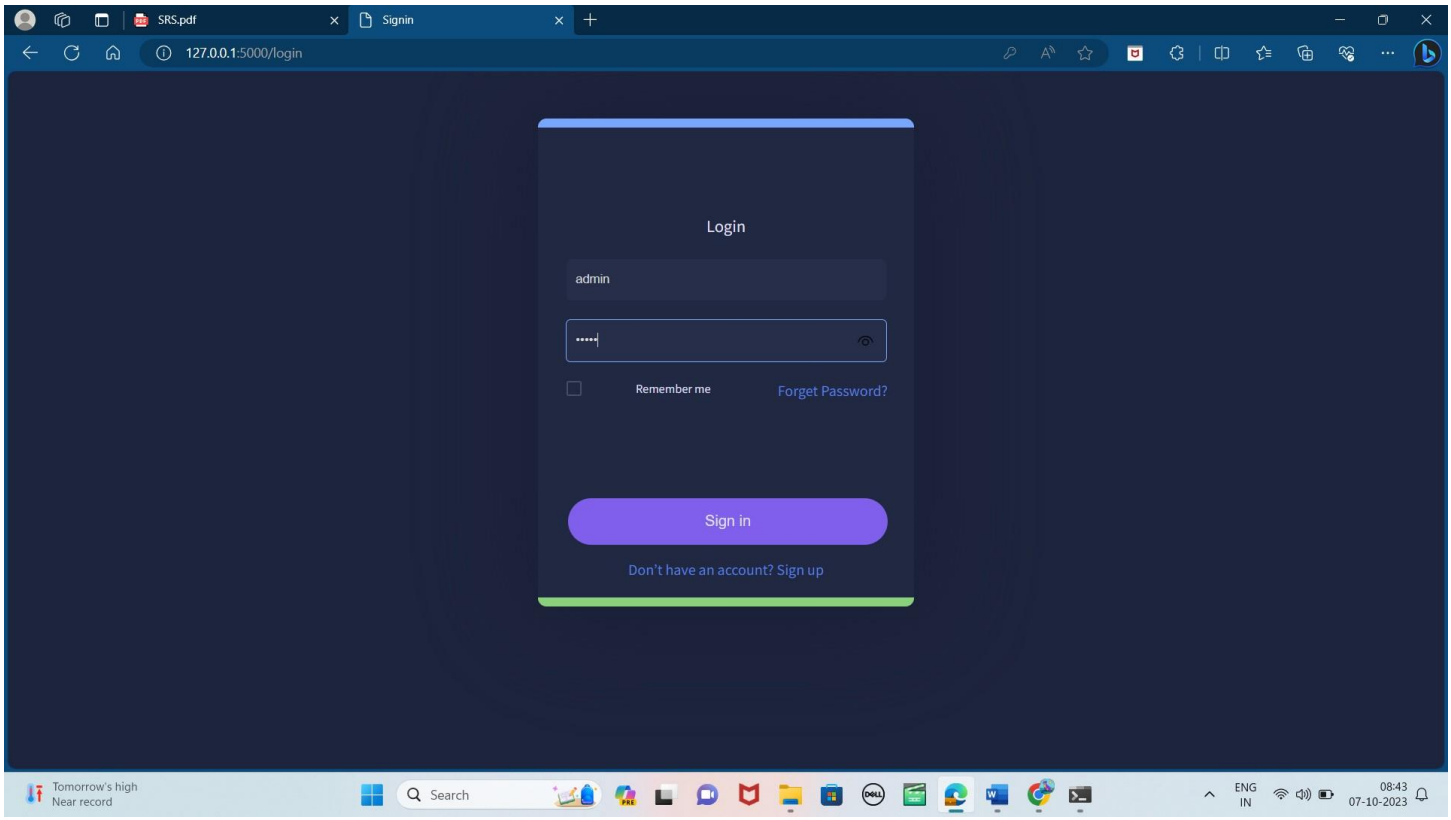
```
# creating the dataframes that will store the evaluation metrics results

df_acc = pd.DataFrame(np.array([dic['acc'] for dic in listofdicts]).T, columns= [dic['modelname'] for dic in listofdicts])

df_fscore = pd.DataFrame(np.array([dic['fscore'] for dic in listofdicts]).T, columns=[dic['modelname'] for dic in listofdicts])

df_recall = pd.DataFrame(np.array([dic['recall'] for dic in listofdicts]).T, columns=[dic['modelname'] for dic in listofdicts])

df_precision = pd.DataFrame(np.array([dic['precision'] for dic in listofdicts]).T, columns=[dic['modelname'] for dic in listofdicts])

df_time = pd.DataFrame(np.array([dic['tempo'] for dic in listofdicts]).T, columns=[dic['modelname'] for dic in listofdicts])

import plotly.graph_objects as go
import numpy as np


def plot_boxplot(df, title):

    fig = go.Figure()
    # Use x instead of y argument for horizontal plot
    fig.add_trace(go.Box(x=df['ADA-B'], name='ADA Boost', marker_color='gray'))
    fig.add_trace(go.Box(x=df['SVMR'], name = 'SVM RBF',  marker_color='gray'))
    fig.add_trace(go.Box(x=df['RF'], name= 'Random Forest',  marker_color='gray'))
    fig.add_trace(go.Box(x=df['LR'], name= 'Logistic Regression',  marker_color='gray'))
    fig.add_trace(go.Box(x=df['MLP'], name='MLP',  marker_color='gray'))

    fig.update_layout(title=title)
    fig.update_layout(template='plotly_white')

    fig.show()

plot_boxplot(df_time, 'Time')
plot_boxplot(df_recall, 'Recall')
plot_boxplot(df_fscore, 'Fscore')

plot_boxplot(df_precision, 'Precision')
plot_boxplot(df_acc, 'Accuracy')
# Making the statistic test to each one of the evaluation metrics
create_statistic_test('f1-score', df_fscore)
print()

create_statistic_test('recall', df_recall)
```

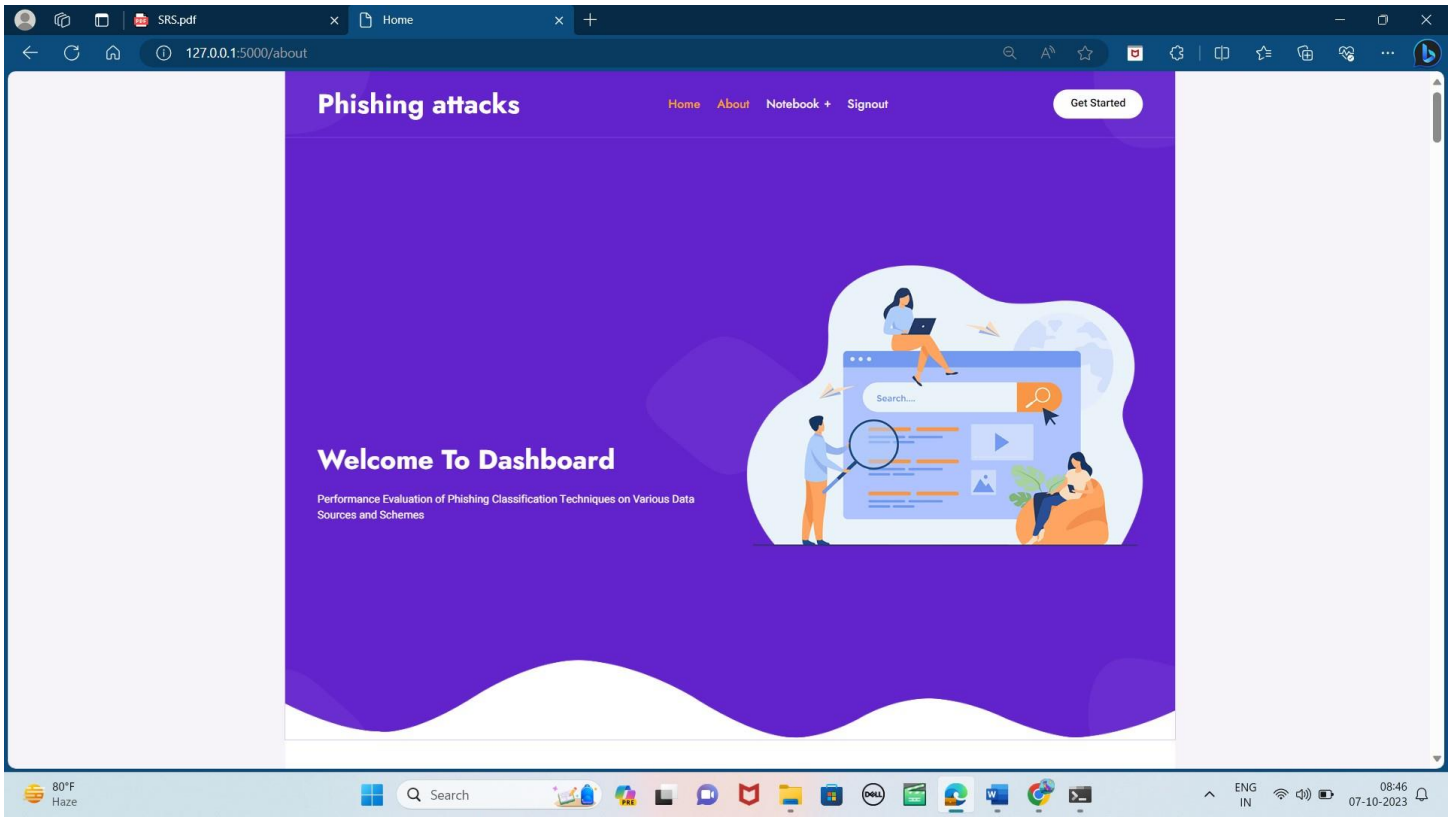
```
print()

create_statistic_test('precision', df_precision)
print()

create_statistic_test('accuracy', df_acc)
```

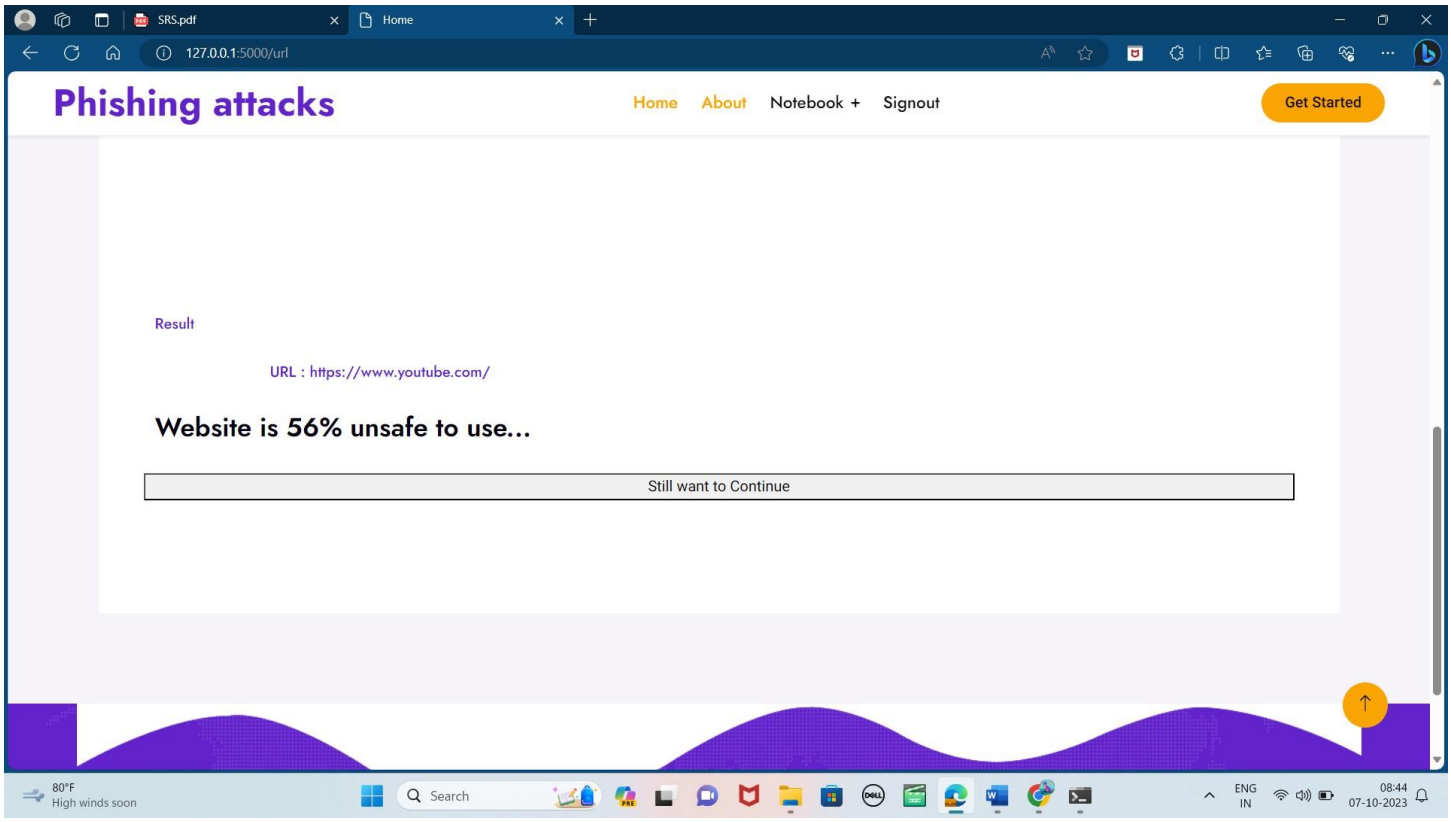
5.2: Output Screens:

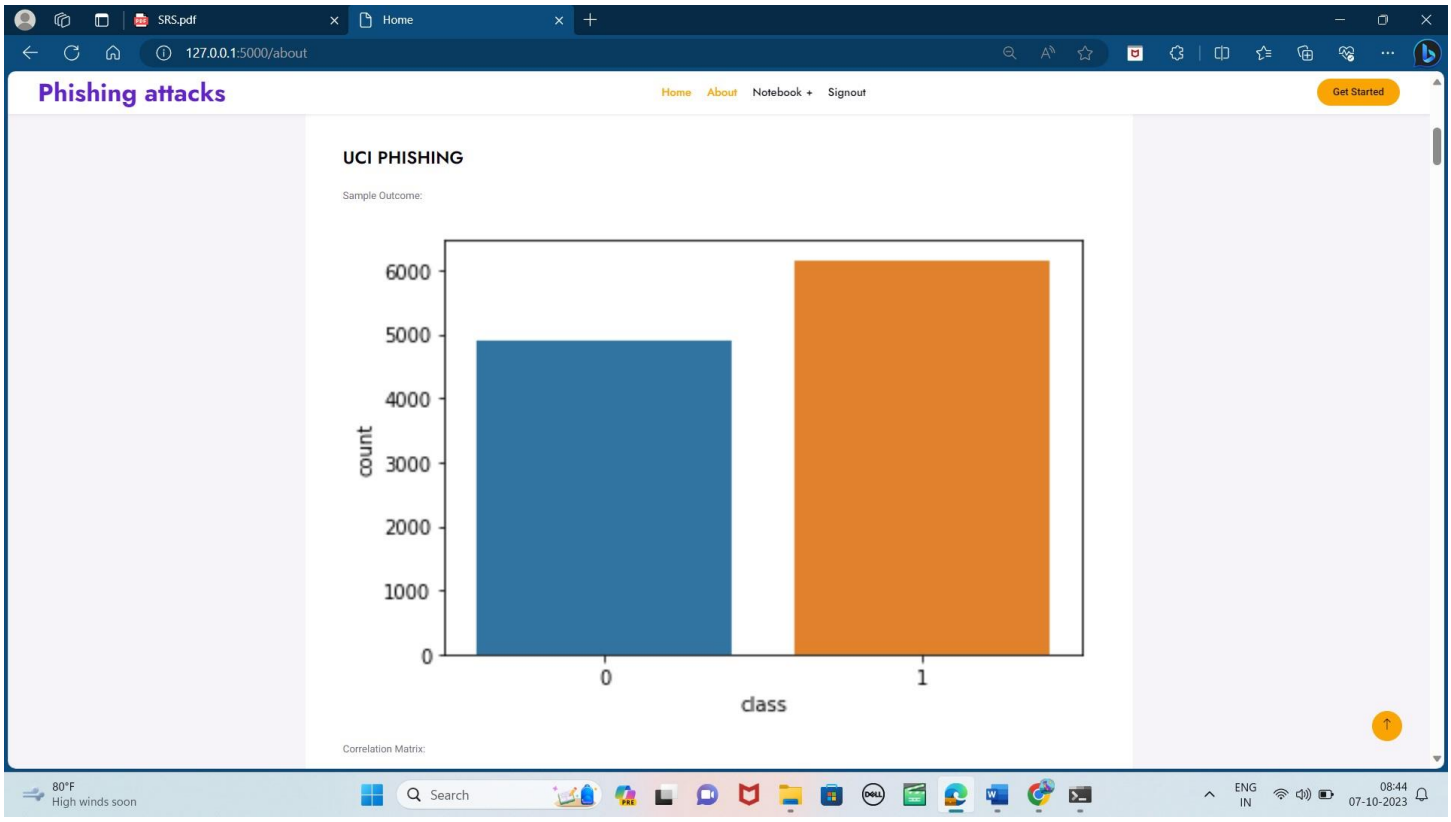


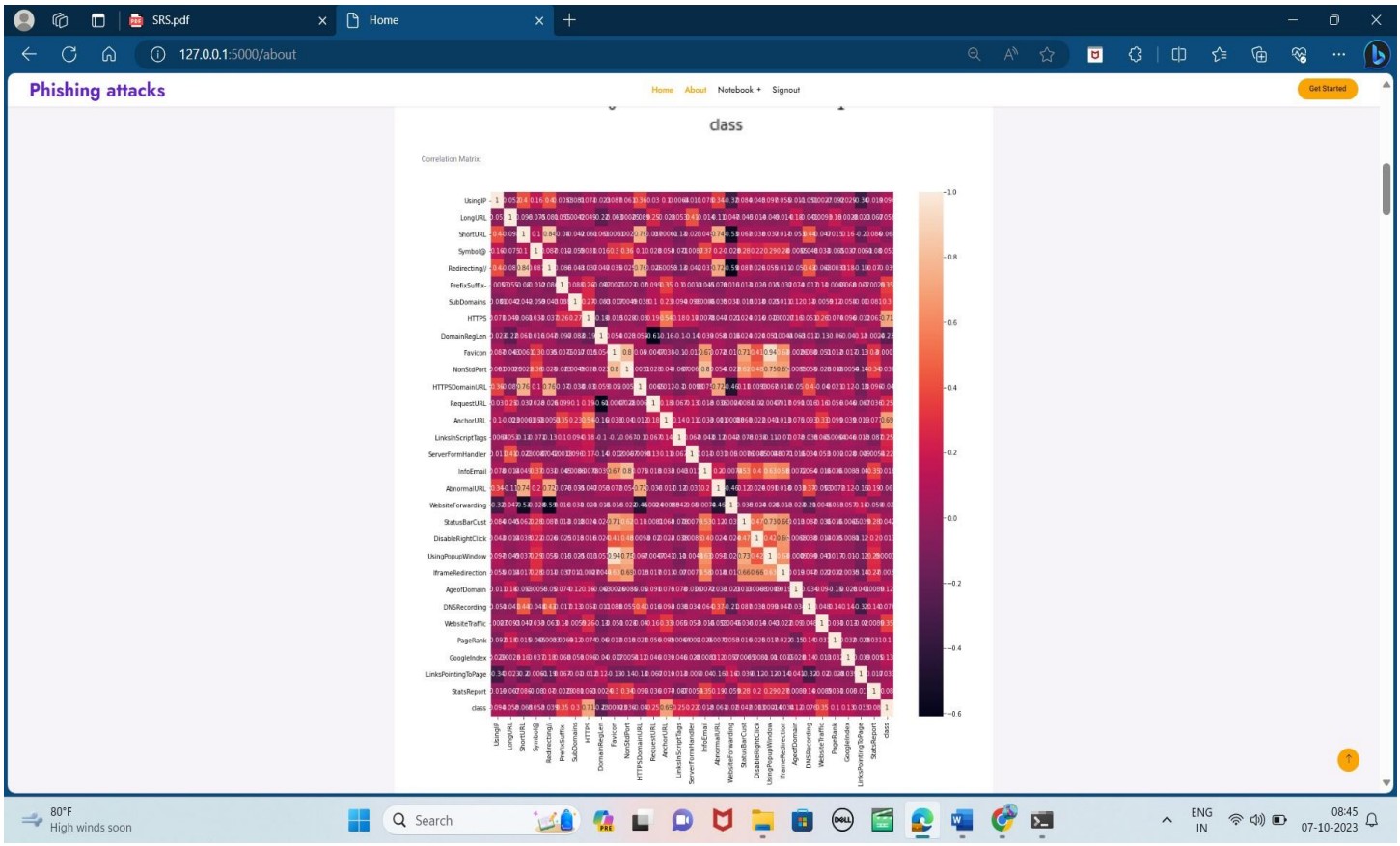



SRS.pdf
Signin
127.0.0.1:5000/login
Login
admin
Remember me
Forget Password?
Sign in
Don't have an account? Sign up
Search


SRS.pdf
Home
127.0.0.1:5000/about
Phishing attacks
Home
About
Notebook +
Signout
Get Started
Welcome To Dashboard
Performance Evaluation of Phishing Classification Techniques on Various Data Sources and Schemes
Search....
Search

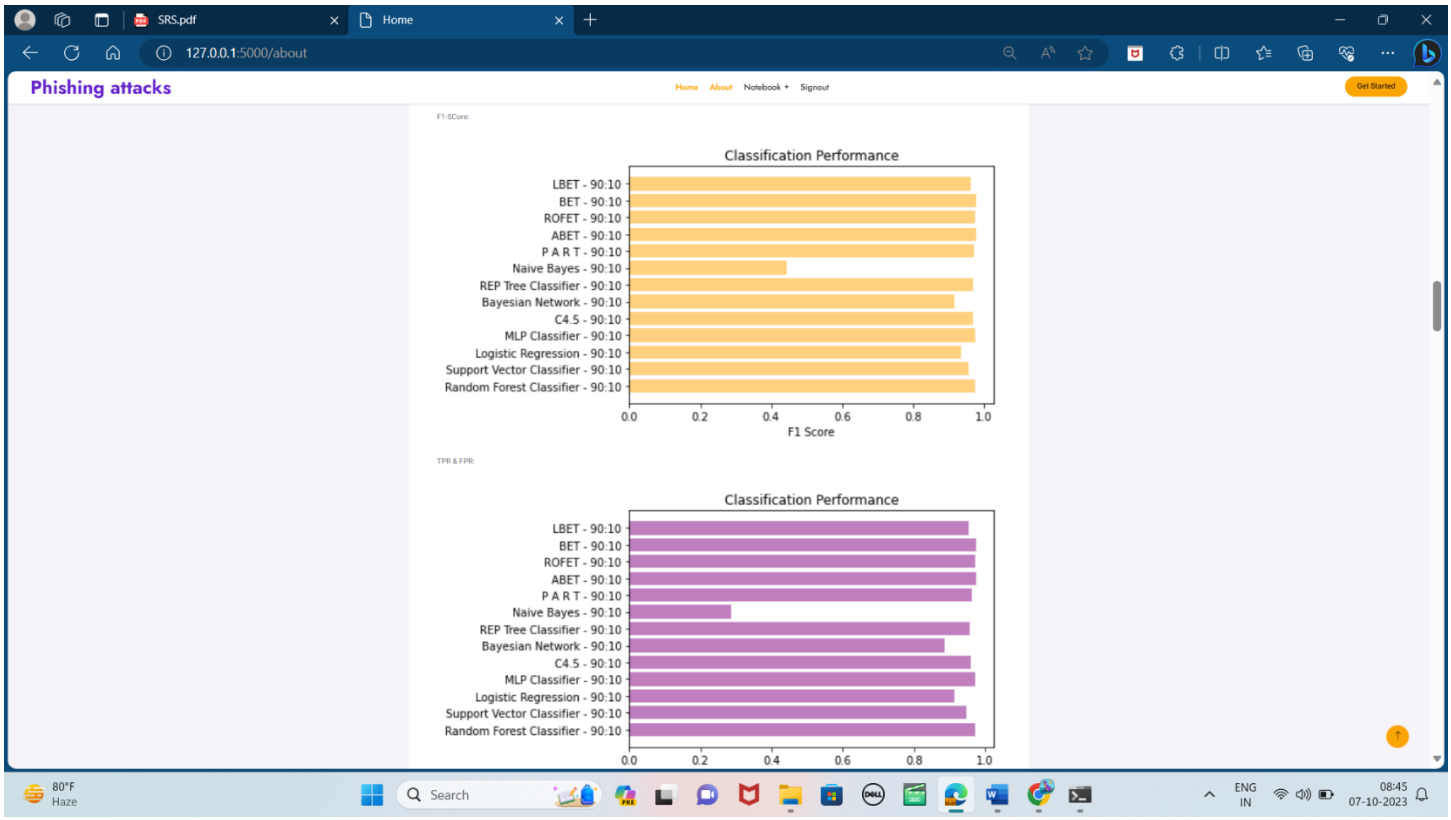

SRS.pdf
Home
127.0.0.1:5000/url
Phishing attacks
Home
About
Notebook +
Signout
Get Started
Result
URL : https://www.youtube.com/
Website is 56% unsafe to use...
Still want to Continue
80°F
High winds soon
Search
ENG
IN
08:44
07-10-2023


SRS.pdf
Home
127.0.0.1:5000/about
Phishing attacks
Home
About
Notebook +
Signout
Get Started
UCI PHISHING
Sample Outcome:
6000
5000
4000
3000
2000
1000
0
count
0
1
class
Correlation Matrix:
80°F
High winds soon
Search
ENG
IN
08:44
07-10-2023


SRS.pdf
Home
127.0.0.1:5000/about
Phishing attacks
Home About Notebook + Signout
Get Started
class
Correlation Matrix:
80°F
High winds soon
Search
ENG
IN
08:45
07-10-2023


SRS.pdf
Home
127.0.0.1:5000/about
Phishing attacks
Home About Notebook + Signout
Get Started
Accuracy:
Classification Performance
LBET - 90:10
BET - 90:10
ROFET - 90:10
ABET - 90:10
P A R T - 90:10
Naive Bayes - 90:10
REP Tree Classifier - 90:10
Bayesian Network - 90:10
C4.5 - 90:10
MLP Classifier - 90:10
Logistic Regression - 90:10
Support Vector Classifier - 90:10
Random Forest Classifier - 90:10
0.0
0.2
0.4
0.6
0.8
1.0
Accuracy Score
Precision:
Classification Performance
LBET - 90:10
BET - 90:10
ROFET - 90:10
ABET - 90:10
P A R T - 90:10
Naive Bayes - 90:10
REP Tree Classifier - 90:10
Bayesian Network - 90:10
C4.5 - 90:10
MLP Classifier - 90:10
Logistic Regression - 90:10
Support Vector Classifier - 90:10
Random Forest Classifier - 90:10
0.0
0.2
0.4
0.6
0.8
1.0
80°F
High winds soon
Search
ENG
IN
08:45
07-10-2023

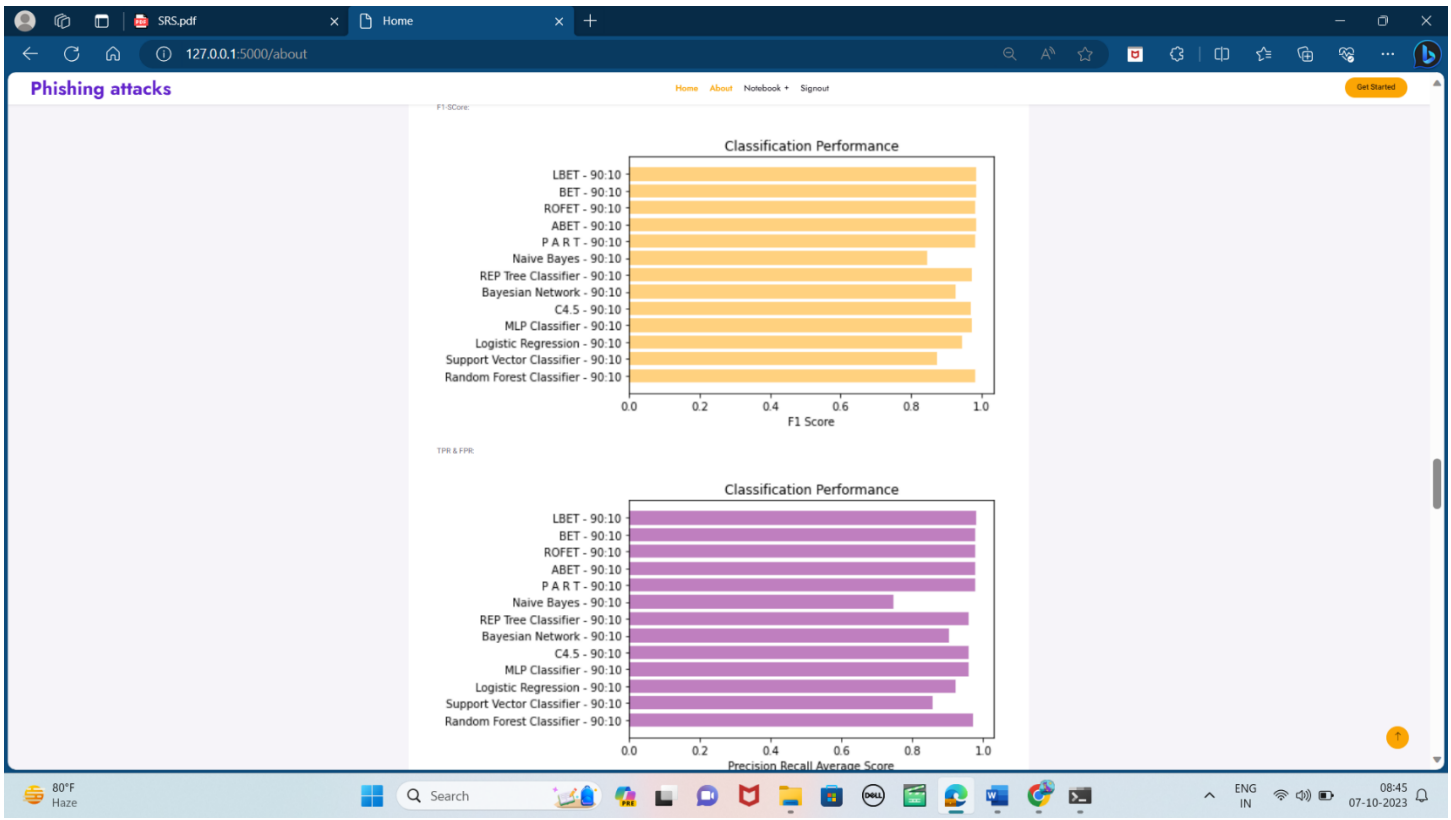

SRS.pdf
Home
127.0.0.1:5000/about
Phishing attacks
Home About Notebook + Signout
Get Started
Precision Score
Recall:
Classification Performance
LBET - 90:10
BET - 90:10
ROFET - 90:10
ABET - 90:10
P A R T - 90:10
Naive Bayes - 90:10
REP Tree Classifier - 90:10
Bayesian Network - 90:10
C4.5 - 90:10
MLP Classifier - 90:10
Logistic Regression - 90:10
Support Vector Classifier - 90:10
Random Forest Classifier - 90:10
0.0 0.2 0.4 0.6 0.8 1.0
Recall Score
F1-SCore:
Classification Performance
LBET - 90:10
BET - 90:10
ROFET - 90:10
ABET - 90:10
P A R T - 90:10
Naive Bayes - 90:10
REP Tree Classifier - 90:10
Bayesian Network - 90:10
C4.5 - 90:10
MLP Classifier - 90:10
Logistic Regression - 90:10
Support Vector Classifier - 90:10
Random Forest Classifier - 90:10
80°F
Haze
Search
ENG
IN
08:45
07-10-2023

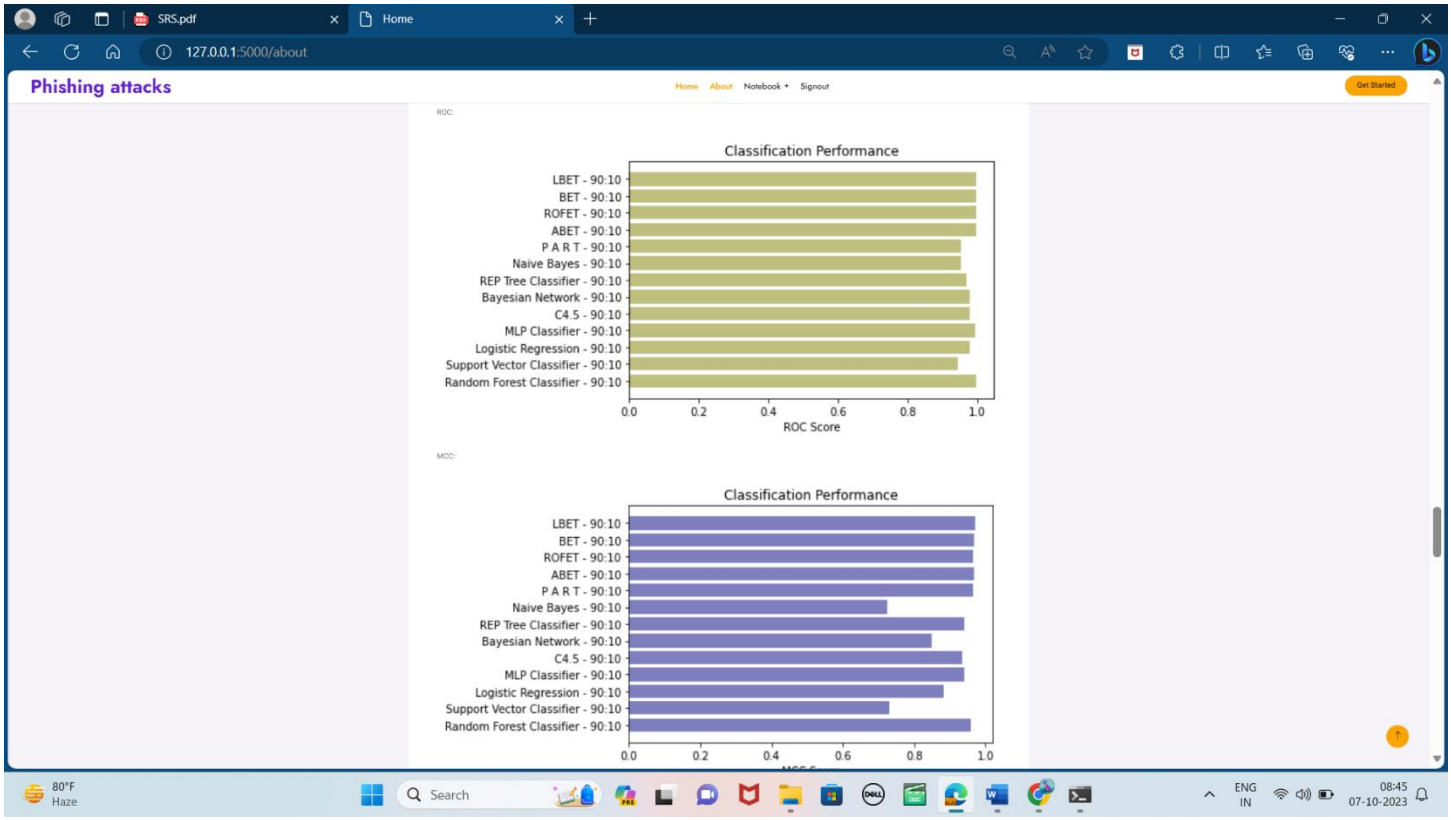

SRS.pdf
Home
127.0.0.1:5000/about
Phishing attacks
Home About Notebook + Signout
Get Started
F1-SCore:
Classification Performance
LBET - 90:10
BET - 90:10
ROFET - 90:10
ABET - 90:10
P A R T - 90:10
Naive Bayes - 90:10
REP Tree Classifier - 90:10
Bayesian Network - 90:10
C4.5 - 90:10
MLP Classifier - 90:10
Logistic Regression - 90:10
Support Vector Classifier - 90:10
Random Forest Classifier - 90:10
0.0 0.2 0.4 0.6 0.8 1.0
F1 Score
TPR & FPR:
Classification Performance
LBET - 90:10
BET - 90:10
ROFET - 90:10
ABET - 90:10
P A R T - 90:10
Naive Bayes - 90:10
REP Tree Classifier - 90:10
Bayesian Network - 90:10
C4.5 - 90:10
MLP Classifier - 90:10
Logistic Regression - 90:10
Support Vector Classifier - 90:10
Random Forest Classifier - 90:10
80°F
Haze
Search
ENG
IN
08:45
07-10-2023


SRS.pdf
Home
127.0.0.1:5000/about
Phishing attacks
Home About Notebook + Signout
Get Started
Classification Performance
LBET - 90:10
BET - 90:10
ROFET - 90:10
ABET - 90:10
P A R T - 90:10
Naive Bayes - 90:10
REP Tree Classifier - 90:10
Bayesian Network - 90:10
C4.5 - 90:10
MLP Classifier - 90:10
Logistic Regression - 90:10
Support Vector Classifier - 90:10
Random Forest Classifier - 90:10
0.0 0.2 0.4 0.6 0.8 1.0
ROC Score
MCC:
Classification Performance
0.0 0.2 0.4 0.6 0.8
MCC Score
80°F Haze
Search
ENG IN
08:45 07-10-2023


SRS.pdf
Home
127.0.0.1:5000/about
Phishing attacks
Home About Notebook + Signout
Get Started
Accuracy:
Classification Performance
LBET - 90:10
BET - 90:10
ROFET - 90:10
ABET - 90:10
P A R T - 90:10
Naive Bayes - 90:10
REP Tree Classifier - 90:10
Bayesian Network - 90:10
C4.5 - 90:10
MLP Classifier - 90:10
Logistic Regression - 90:10
Support Vector Classifier - 90:10
Random Forest Classifier - 90:10
0.0 0.2 0.4 0.6 0.8 1.0
Accuracy Score
Precision:
Classification Performance
0.0 0.2 0.4 0.6 0.8 1.0
80°F Haze
Search
ENG IN
08:45 07-10-2023


SRS.pdf
Home
127.0.0.1:5000/about
Phishing attacks
Home About Notebook + Signout
Get Started
Classification Performance
LBET - 90:10
BET - 90:10
ROFET - 90:10
ABET - 90:10
P A R T - 90:10
Naive Bayes - 90:10
REP Tree Classifier - 90:10
Bayesian Network - 90:10
C4.5 - 90:10
MLP Classifier - 90:10
Logistic Regression - 90:10
Support Vector Classifier - 90:10
Random Forest Classifier - 90:10
0.0 0.2 0.4 0.6 0.8 1.0
Precision Score
Recall:
Classification Performance
Recall Score
80°F Haze
Search
ENG IN
08:45 07-10-2023

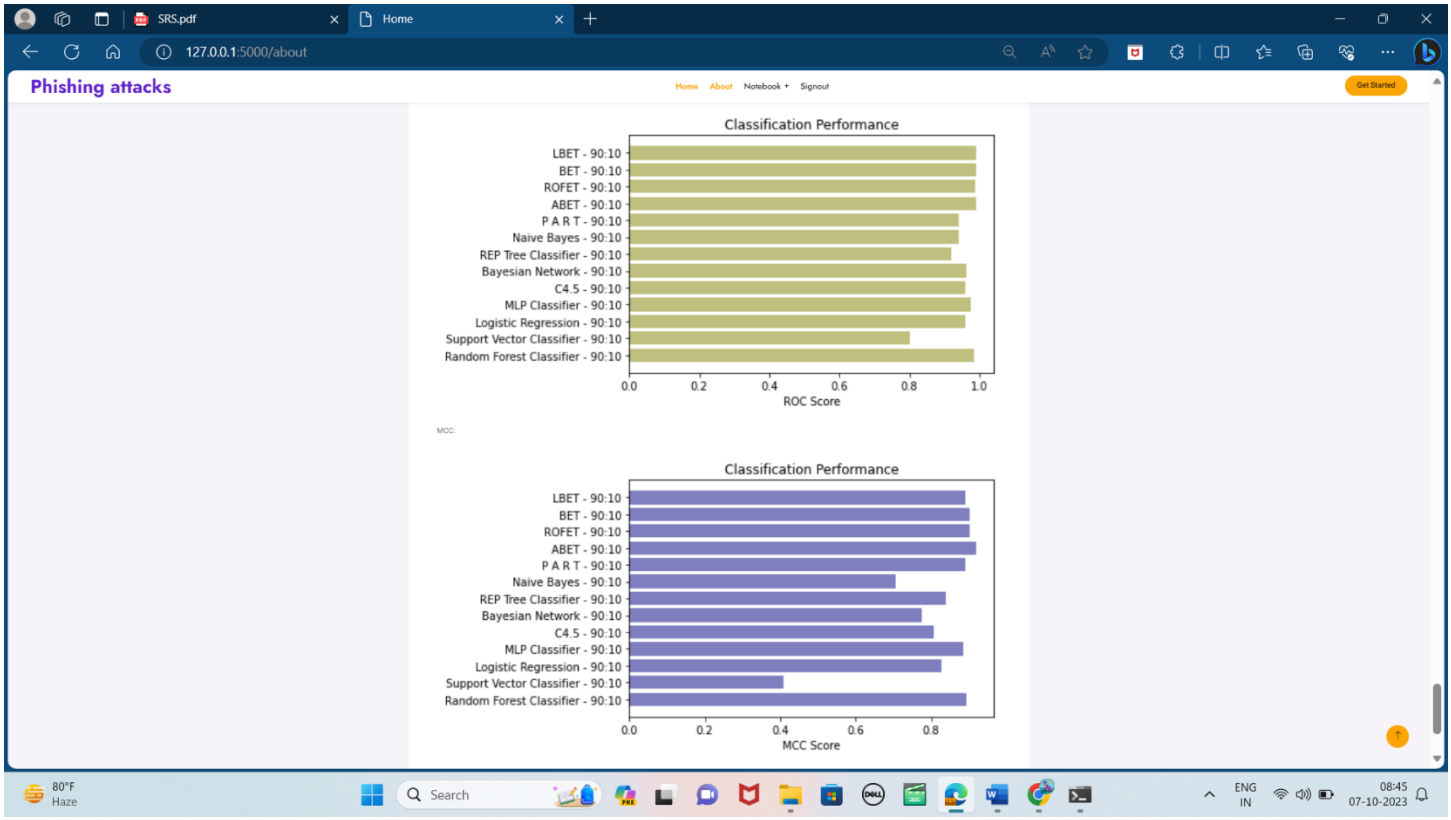

SRS.pdf
Home
127.0.0.1:5000/about
Phishing attacks
Home About Notebook + Signout
Get Started
F1-SCore:
Classification Performance
LBET - 90:10
BET - 90:10
ROFET - 90:10
ABET - 90:10
P A R T - 90:10
Naive Bayes - 90:10
REP Tree Classifier - 90:10
Bayesian Network - 90:10
C4.5 - 90:10
MLP Classifier - 90:10
Logistic Regression - 90:10
Support Vector Classifier - 90:10
Random Forest Classifier - 90:10
0.0 0.2 0.4 0.6 0.8 1.0
F1 Score
TPR & FPR:
Classification Performance
Precision Recall Average Score
80°F Haze
Search
ENG IN
08:45 07-10-2023


SRS.pdf
Home
127.0.0.1:5000/about
Phishing attacks
Home About Notebook + Signout
Get Started
ROC:
Classification Performance
LBET - 90:10
BET - 90:10
ROFET - 90:10
ABET - 90:10
P A R T - 90:10
Naive Bayes - 90:10
REP Tree Classifier - 90:10
Bayesian Network - 90:10
C4.5 - 90:10
MLP Classifier - 90:10
Logistic Regression - 90:10
Support Vector Classifier - 90:10
Random Forest Classifier - 90:10
0.0 0.2 0.4 0.6 0.8 1.0
ROC Score
MCC:
Classification Performance
LBET - 90:10
BET - 90:10
ROFET - 90:10
ABET - 90:10
P A R T - 90:10
Naive Bayes - 90:10
REP Tree Classifier - 90:10
Bayesian Network - 90:10
C4.5 - 90:10
MLP Classifier - 90:10
Logistic Regression - 90:10
Support Vector Classifier - 90:10
Random Forest Classifier - 90:10
0.0 0.2 0.4 0.6 0.8 1.0
80°F Haze
Search
ENG IN
08:45 07-10-2023


SRS.pdf
Home
127.0.0.1:5000/about
Phishing attacks
Home About Notebook + Signout
Get Started
MCC Score
UCI SPAMBASE
Accuracy:
Classification Performance
LBET - 90:10
BET - 90:10
ROFET - 90:10
ABET - 90:10
P A R T - 90:10
Naive Bayes - 90:10
REP Tree Classifier - 90:10
Bayesian Network - 90:10
C4.5 - 90:10
MLP Classifier - 90:10
Logistic Regression - 90:10
Support Vector Classifier - 90:10
Random Forest Classifier - 90:10
0.0 0.2 0.4 0.6 0.8 1.0
Accuracy Score
Precision:
Classification Performance
LBET - 90:10
BET - 90:10
ROFET - 90:10
ABET - 90:10
P A R T - 90:10
Naive Bayes - 90:10
REP Tree Classifier - 90:10
Bayesian Network - 90:10
C4.5 - 90:10
MLP Classifier - 90:10
Logistic Regression - 90:10
80°F Haze
Search
ENG IN
08:45 07-10-2023


SRS.pdf
Home
127.0.0.1:5000/about
Phishing attacks
Home About Notebook + Signout
Get Started
Classification Performance
LBET - 90:10
BET - 90:10
ROFET - 90:10
ABET - 90:10
P A R T - 90:10
Naive Bayes - 90:10
REP Tree Classifier - 90:10
Bayesian Network - 90:10
C4.5 - 90:10
MLP Classifier - 90:10
Logistic Regression - 90:10
Support Vector Classifier - 90:10
Random Forest Classifier - 90:10
0.0 0.2 0.4 0.6 0.8 1.0
Precision Score
Recall:
Classification Performance
Recall Score
F1-SCore:
80°F Haze
Search
ENG IN
08:45 07-10-2023


SRS.pdf
Home
127.0.0.1:5000/about
Phishing attacks
Home About Notebook + Signout
Get Started
Classification Performance
LBET - 90:10
BET - 90:10
ROFET - 90:10
ABET - 90:10
P A R T - 90:10
Naive Bayes - 90:10
REP Tree Classifier - 90:10
Bayesian Network - 90:10
C4.5 - 90:10
MLP Classifier - 90:10
Logistic Regression - 90:10
Support Vector Classifier - 90:10
Random Forest Classifier - 90:10
0.0 0.2 0.4 0.6 0.8 1.0
F1 Score
TPR & FPR:
Classification Performance
Precision Recall Average Score
80°F Haze
Search
ENG IN
08:45 07-10-2023



5.3: Testing

Software testing is a critical element of software quality assurance and represents the ultimate review of specification, design and code generation.

CHAPTER 6
CONCLUSION & FUTURE SCOPE

The conclusion of this product is that it is an efficient and robust method for phishing detection using the best classification techniques. The system developed is cost-efficient and needs less installation. The product's users tend to produce a significant increase and decrease with respect to performance. The findings of this research prove that unused data significantly affects performance during the classification process. There was a significant performance increase and decrease in the subset scheme. In future, this subset scheme needs to be applied to confirmed cases to compare its performances. Moreover, certain recommendations were proposed for developing research on improving phishing classification techniques.

CHAPTER 7
REFERENCES

[1] R. S. Rao and A. R. Pais, ‘‘Detection of phishing websites using an efficient feature-based machine learning framework,’’ Neural Comput. Appl., vol. 31, no. 8, pp. 3851–3873, Aug. 2019.

[2] K. L. Chiew, C. L. Tan, K. Wong, K. S. C. Yong, and W. K. Tiong, ‘‘A new hybrid ensemble feature selection framework for machine learning-based phishing detection system,’’ Inf. Sci., vol. 484, pp. 153–166, May 2019.

[3] O. K. Sahingoz, E. Buber, O. Demir, and B. Diri, ‘‘Machine learning based phishing detection from URLs,’’ Expert Syst. Appl., vol. 117, pp. 345–357, Mar. 2019.

[4] S. W. Liew, N. F. M. Sani, M. T. Abdullah, R. Yaakob, and M. Y. Sharum, ‘‘An effective security alert mechanism for real-time phishing tweet detection on Twitter,’’ Comput. Secur., vol. 83, pp. 201–207, Jun. 2019.

[5] V. Muppavarapu, A. Rajendran, and S. K. Vasudevan, ‘‘Phishing detection using RDF and random forests,’’ Int. Arab J. Inf. Technol., vol. 15, no. 5, pp. 817–824, 2018.

[6] A. S. Bozkir and M. Aydos, ‘‘LogoSENSE: A companion HOG based logo detection scheme for phishing web page and E-mail brand recognition,’’ Comput. Secur., vol. 95, Aug. 2020, Art. no. 101855.

[7] S. E. Raja and R. Ravi, ‘‘A performance analysis of software defined network based prevention on phishing attack in cyberspace using a deep machine learning with CANTINA approach (DMLCA),’’ Comput. Commun., vol. 153, pp. 375–381, Mar. 2020.

[8] M. Sameen, K. Han, and S. O. Hwang, ‘‘PhishHaven—An efficient real-time AI phishing URLs detection system,’’ IEEE Access, vol. 8, pp. 83425–83443, 2020.

[9] R. S. Rao, T. Vaishnavi, and A. R. Pais, ‘‘PhishDump: A multi-model ensemble based technique for the detection of phishing sites in mobile devices,’’ Pervasive Mobile Comput., vol. 60, Nov. 2019, Art. no. 101084.

[10] Y. Ding, N. Luktarhan, K. Li, and W. Slamu, ‘‘A keyword-based combination approach for detecting phishing webpages,’’ Comput. Secur., vol. 84, pp. 256–275, Jul. 2019.

[11] A. K. Jain and B. B. Gupta, ‘‘A machine learning based approach for phishing detection using hyperlinks information,’’ J. Ambient Intell. Hum. Comput., vol. 10, no. 5, pp. 2015–2028, May 2019.

[12] A. E. Aassal, S. Baki, A. Das, and R. M. Verma, ‘‘An in-depth benchmarking and evaluation of phishing detection research for security needs,’’ IEEE Access, vol. 8, pp. 22170–22192, 2020.

[13] P. Vaitkevicius and V. Marcinkevicius, ‘‘Comparison of classification algorithms for detection of phishing websites,’’ Informatica, vol. 31, no. 1, pp. 143–160, Mar. 2020.

[14] Y.-H. Chen and J.-L. Chen, ‘‘AI@ntiPhish—Machine learning mechanisms for cyber-phishing attack,’’ IEICE Trans. Inf. Syst., vol. E102.D, no. 5, pp. 878–887, May 2019.

[15] C. L. Tan, K. L. Chiew, K. S. C. Yong, S. N. Sze, J. Abdullah, and Y. Sebastian, ‘‘A graph-theoretic approach for the detection of phishing webpages,’’ Comput. Secur., vol. 95, Aug. 2020, Art. no. 101793.