A MINI PROJECT

ON

DETECTION PHISHING WEBSITES FROM URLs

Submitted to

JAWARALAL NEHRU TECNOLOGICAL UNIVERSITY, HYDERABAD

In partial fulfilment of the requirements for the award of Post Graduation of

MASTER

IN

COMPUTER APPLICATIONS

submitted By

CHANDRAGIRI GAYATHRI **23UK1F0002**

Under the guidance of

Mr. P .Ilanna

Assistant professor

DEPARTMENT OF MASTER OF COMPUTER APPLICATIONS

VAAGDEVI ENGINEERING COLLEGE

Affiliated to JNTUH, Hyderabad

Bollikunta, Warangal (TS)

506005

DEPARTMENT OF

MASTER OF COMPUTER APPLICATIONS

VAAGDDEVI ENGINEERING COLLEGE(WARANGAL)

CERTIFICATE OF COMPLETION

INDUSTRY ORIENTED MINI PROJECT

This is to certify that Project work entitled "DETECTION OF PHISHING WEBSITES FROM URLs" is being submitted by CHANDRAGIRI GAYATHRI in partially fulfilment of the requirements for the award of postgraduate of Master of Computer Applications to Jawaharlal Nehru Technological University Hyderabad during the academic year 2023-2024

Project Guide	**HOD**
Mr. P. Ilanna	**Dr . R. Naveen kumar**
Assistant professor	

ACKNOWLEDGEMENT

I wish to take this opportunity to express our sincere gratitude and deep sense of respect to our beloved **Dr. Syed Musthak Ahmed,** Principle, Vaagdevi Engineering College for making me available all the required assistance and for his support and inspiration to carry out this project working the institute.

I extend our heartfelt thanks to Dr. R.NAVEEN KUMAR ,Head of department of Master of computer Applications ,Vaagdevi Engineering College for providing me necessary infrastructure and thereby giving us freedom to carry out the project work.

I express heartfelt thanks to Smart Bridge Educational Services Private Limited, for their constant supervision as well as for providing me necessary information regarding the Project work and for their support in completing the project work.

I express heartfelt thanks to the guide, P. ILANNA, for his constant support and giving necessary guidance for completion of this project work.

Finally, I express our sincere thanks and gratitude to my family members, friends for their encouragement and outpouring their knowledge and experience throughout the thesis.

CHANDRAGIRI GAYATHRI 23UK1F0002

ABSTRACT

Phishing attacks pose a significant threat to online security, targeting individuals and organizations with the intent of stealing information such as usernames, passwords, financial details. As these attacks become increasingly sophisticated, traditional methods of detection of phishing websites. Leveraging the power of computational models to identify subtle patterns indicative of malicious intent. The study begins by collecting a diverse dataset of legitimate and phishing websites, encompassing various features such as website content, structure, and user interaction patterns. Features engineering plays a crucial role in extracting relevant information from these diverse datapoints, facilitating the training of machine learning models. Various algorithms, including logistic regression and random forest

TABLE OF CONTENTS

1.INTRODUCTION

In the ever -expanding digital landscape, the proliferation of online services and transactions has given rise to a parallel surge in cyber threats, with phishing attacks standing out as a pervasive and sophisticated menace. Phishing, the deceptive practice of luring individuals into divulging sensitive information by masquerading as a trustworthy entity, poses a severe threat to the security and privacy of users. Traditional methods of detecting phishing websites have proven insufficient in the face of evolving tactics employed by malicious actors. As a result, the integration of advanced technologies, particularly machine learning techniques, has emerged as a promising avenue to fortify the defence again these clandestine threats. phishing attacks often involve the creation of fraudulent websites that closely mimic legitimate platform, making it challenging for users to distinguish between the genuine and the malicious. Machine learning, with its capacity to discern intricate patterns and relationships within vast dataset, offers a potent tool for the automated identification of phishing websites. This research endeavors to delve into the realm of phishing website detection, focusing on the application of diverse machine learning algorithms to effectively discriminate between authentic and deceptive online entities.

The motivation for this research stems from the critical need to bolster cybersecurity measures and safeguard sensitive information in an era where online interactions have become integral to daily life. By harvesting the power of machine learning, this study seeks to contribute to the development of proactive and adaptive systems capable of staying ahead of the dynamic tactics employed by cybercriminals. Through a comprehensive exploration of the existing literature, methodologies, and emerging trends in the field, this research aims to provide valuable insights into the efficacy and challenges associated with employing machine learning techniques for phishing websites detection. In the following sections, we will delve into the key objectives of the study, the significance of utilizing machine learning in this context.

1.1 EXISTING SYSTEM

Heuristic-Based System

Some systems use heuristic rules to identify potential phishing websites based on known patterns and characteristics associated with phishing attacks. These rules may involve analysing URLs, checking for HTTPS usage, and examining the structure of web pages

Machine Learning Models

Various machine learning algorithms and models are employed for phishing website detection. Common models include decision tree, random forests, support vector machines (SVM), and neural networks. These models are trained on features extracted from datasets containing both legitimate and phishing websites.

Ensemble Methods

Ensemble methods, such as bagging and boosting, are utilized to enhance the performance and robustness of machine learning models. Multiple models are combined to improve accuracy and reduce overfitting.

Cloud Based Solutions

Cloud based solutions provide scalable and centralized phishing detection service. These systems often use machine learning algorithm to analyze large dataset of web traffic and identify potential threats.

Open-source Tools

Several open-source tools and frameworks are available for building and deploying phishing website detection models. These tools may include libraries for machine learning, featuring extraction, and data preprocessing.

Real-Time-Detection System

Many systems are designed for real-time detection, allowing hem to analyze websites as users access them. Real-time detection is critical for providing immediate warning to users and preventing interactions with potential phishing sites

Security measurements

Secure communication systems prioritize secure communication between components to prevent unauthorized access. Access control are implemented to manage permission and restrict unauthorized interactions with the system

Proofpoint TAP

Proofpoint TAP uses machine learning and behaviour analysis to detect and block advances phishing attacks. It analyses URLs, attachments, and other email elements to identify malicious content.

1.2 Proposed system

Dynamic Feature selection: Dynamic feature selection in the context of phishing website detection using machine using machine learning involves adaptively choosing the most relevant feature during the model's operation. This process is particularly beneficial when dealing with evolving threats and changing characteristics of phishing websites implement a dynamic feature selection mechanism that adapts to changing characteristics of phishing attacks. Explore techniques such as recursive feature elimination or information gain to identify the most relevant features for detection.

1.Evolving phishing tactics: Phishing tactics change over time, and attackers may modify the features that distinguish phishing websites. Dynamic features selection allows the model to adopt to these charges.

2.Reducing Dimensionality: As the number of features can be large dynamic selection help in reducing dimensionality ,preventing overfitting , and improving computational efficiency.

3.Enhancing model generalization: By focusing on the most relevant features for the current dataset, dynamic selection can enhance the model's generalization to new and unseen phishing patterns.

4.hybrid machine learning models: Develop a hybrid machine learning model that combine the strengths of different algorithms. consider ensemble methods such as random forests, logistic regression to imporove overall detection accuracy and robustness.

5. continuous learning and updating:

a. scheduled updates Regularly update the machine learning model based on new data to ensure that it remains effective against evoliving phishing tactics.

b. Automated pipeline: An automated pipeline in the context of phishing website detection using machine learning involves a systematic and automated workflow that streamlines and efficient process for continuous learning, model improvement, and adaptation to emerging threats. An automated pipelines ensures that the phishing detection system operates efficiently, adapts to changes in the threat landscape, and continuously improves over time. Regular monitoring and automation reduce manual intervention, making the system more responsive and robust.

1.3 HARDWARE REQUIREMENTS

The hardware requirements for a phishing website detection system using machine learning techniques can vary based on factors such as the size of the dataset, complexity of the machine learning models, and desired level of real-time detection

- Computation resources
- GPU Acceleration
- Storage
- Security measures
- Network Infrastructure

1.4 SOFTWARE REQUIREMENTS

The software requirements for a phishing website detection system using machine learning techniques cover a range of components, from development tools to runtime environments.

Integrated development environment (IDE): choose a suitable IDE for developing machine learning models. Jupyter notebook

Programming language:

Python

2.DESING OF PROJECT

2.1 THEORITICAL ANALYSIS

This project focuses on detecting phishing websites from URLs. By utilizing machine learning algorithms and data analytics, the system aims to analyse website URLs in real-time to identify and flag potential phishing attempts, thereby enhancing online security and protecting user from cyber threats

Scenarios:

Scenario 1: Suspicious Link Detection

Situation: Users receive an email containing a link purportedly from their bank, requesting them to update their account information urgently.

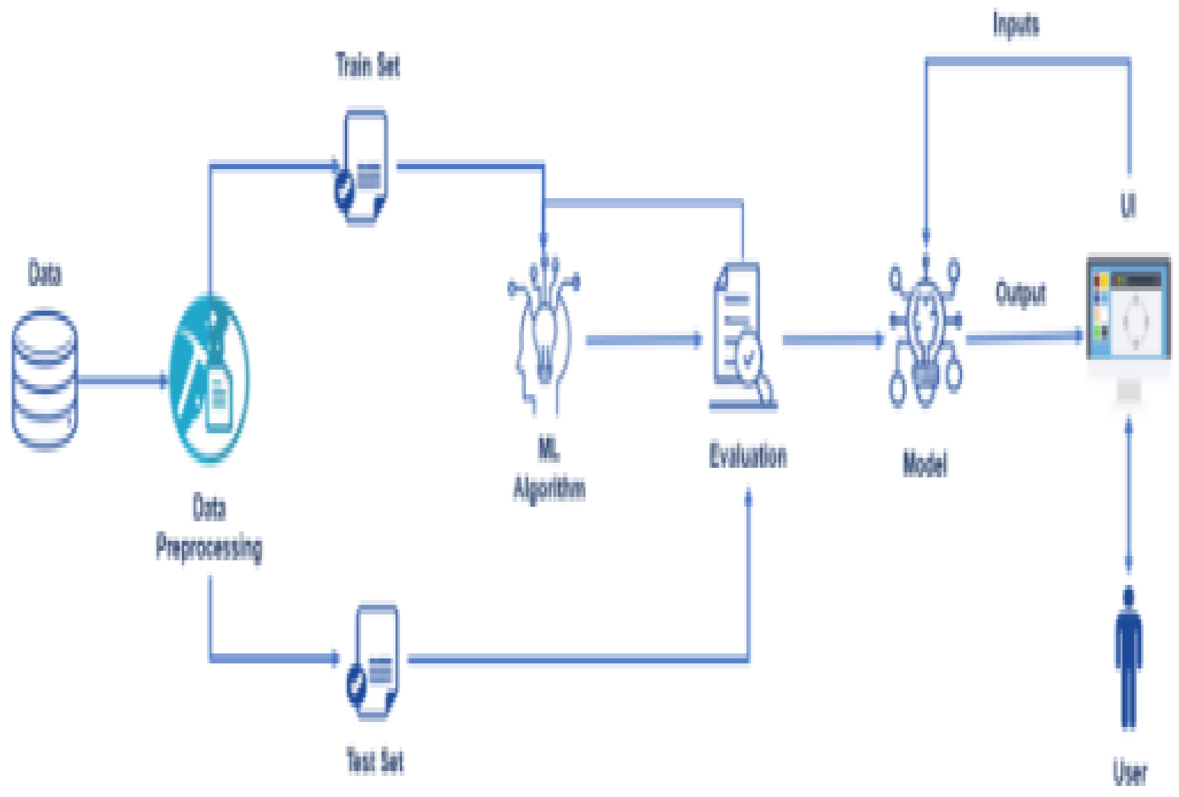
Scenario 2: Social media phishing alert

Situation: A Social media post offers a lucrative discount on a popular product, redirecting users to a website asking for personal information.

Scenario 3: Fake login page identification

Situation 3: Users land on a website that mimics a legitimate login page for a popular online service, attempting to steal login credentials.

2.2 BLOCK DIAGRAM



2.3 SOFTWARE DESIGNING

The following is the software required to complete this project:

- **Jupyter notebook:** Jupyter will serve as the development and execution environment for your predictive modelling, data preprocessing, and model training tasks. It provides a cloud-based environment with access to python libraries and hardware acceleration.

- **Dataset(CSV file):** The dataset in CSV format is essential for training and testing your predictive model. It includes labeled example of legitimate and phishing websites.

- **Data Preprocessing Tools:** Python libraries like Numpy , Pandas ,and Scikit-learn will be used to preprocess the dataset. This includes clean and preprocess the data to handle missing values, normalize feature, and ensure data quality. Transform raw data into a format suitable for machine learning

- **Feature selection /Drop:** Feature selection or dropping unnecessary features from the dataset can be done using Scikit-learner to enhance model efficiency

- **Feature Extractions:** Extract relevant features from the pre-processed data. Features may include URL structure, domain information, SSL certificates, content characteristics and user behaviour.

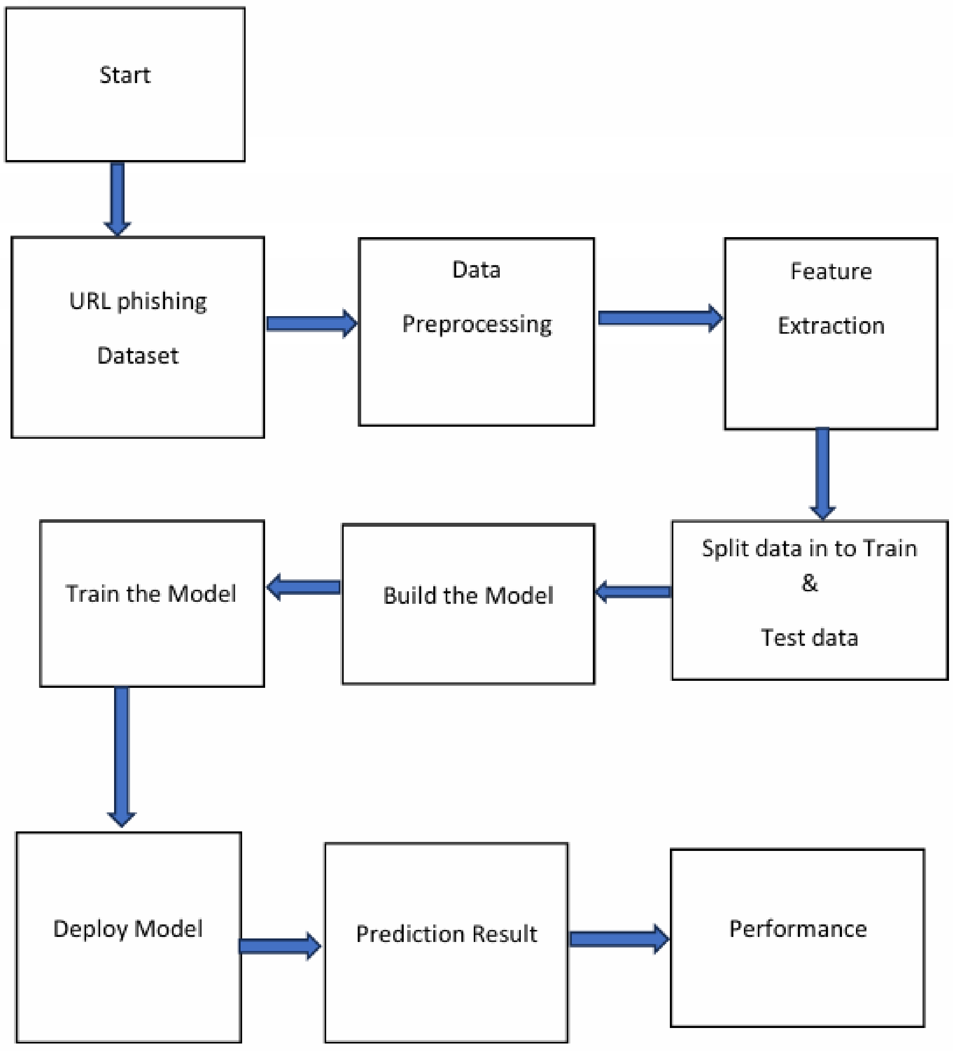
- **Machine learning Model Training:** Implement dynamic feature selection techniques to adaptively choose the most relevant features during model training and runtime. Train the model using the labeled dataset, optimizing hyperparameters for performance validate the model using a separate testing dataset to assess generalization capabilities evaluate performance metrics such as accuracy, precision, recall, and F1 score

- **Deployment the model:** Deploying a machine learning model for phishing website detection involves integrating the trained model into a production environment where it can make real-time predictions.
- **UI Based on Flask Environment:** Flask, a python web framework, will be used to develop the user interface(UI) for the system. Flask application will provide a user-friendly platform for users to input location data.
- Jupyter will be central hub for model development and training , while Flask will facilitate user interaction and data presentation. The dataset , along with data preprocessing , will ensure the quality of the training data, and feature selection will be optimize the model. Finally, model accuracy evaluation will confirm the system’s predictive capabilities.

- **Prediction Result:** The prediction result in a phishing website detection system using machine learning techniques is the output generated by the trained model when it evaluates a website based on its feature. The result indicates whether the website is classified as legitimate or potentially a phishing site.

- **Performance:** The performance of a phishing website detection system using machine learning techniques is typically evaluated using various metrics that measure the effectiveness of the model in distinguish between legitimate and phishing websites.

2.4 FLOWCHART



3.IMPLEMENTATION

Milestone 1: Data Collection & preparation

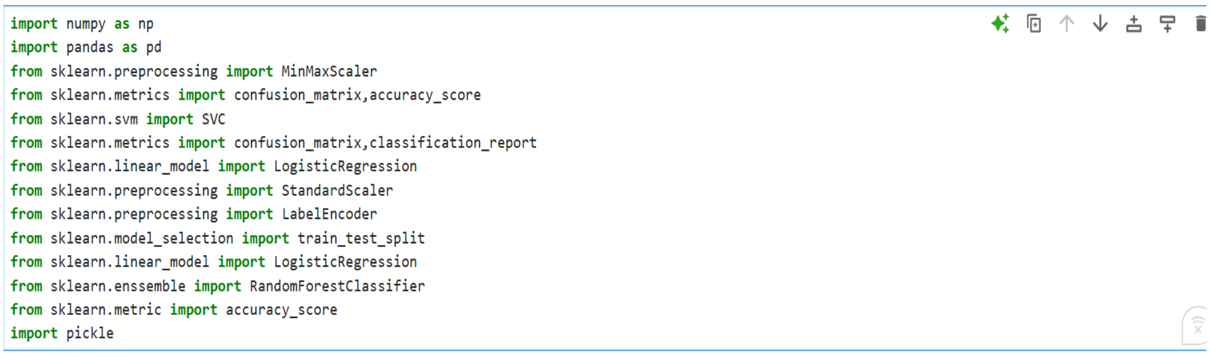
Activity 1: Collecting the data set

There are many popular open sources for collecting the data .eg.kaggle.com ,UCI, repository etc.

In this project I have used .csv data.

Activity 2: Importing the libraries

Importing the necessary libraries as shown in the below images.

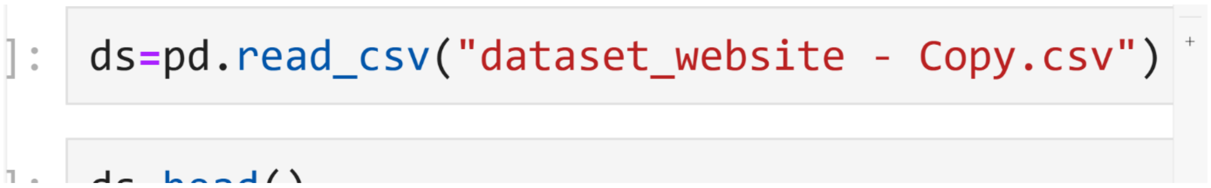


```
import numpy as np
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import confusion_matrix,accuracy_score
from sklearn.svm import SVC
from sklearn.metrics import confusion_matrix,classification_report
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.enssemble import RandomForestClassifier
from sklearn.metric import accuracy_score
import pickle
```

Activity 3:Read Dataset

Our dataset format might be in.csv, excel files, txt, json, etc. we can read the dataset with the help of pandas .

In pandas , we have a function called read_csv() to read the dataset. As a parameter, we have to the directory of the CSV file.



```
]: ds=pd.read_csv("dataset_website - Copy.csv")
```

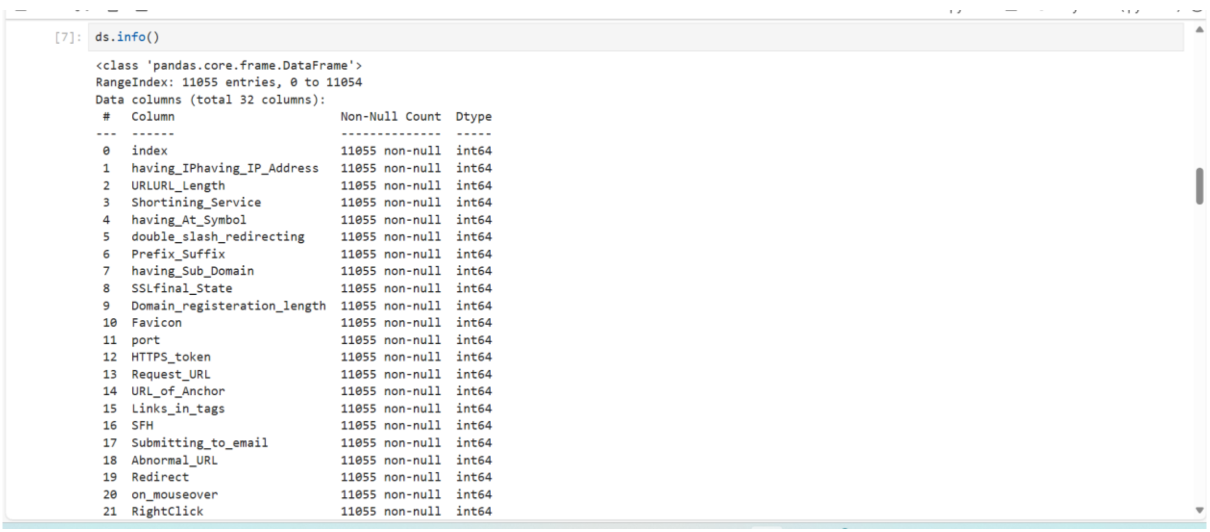
Activity 4:Data preprocessing

As we have understood how the data is, let's pre-process the collected data. The download data set is not suitable for training the machine learning model as it might have so much randomness so we need to clean the dataset properly to fetch good results. This activity includes the following steps.

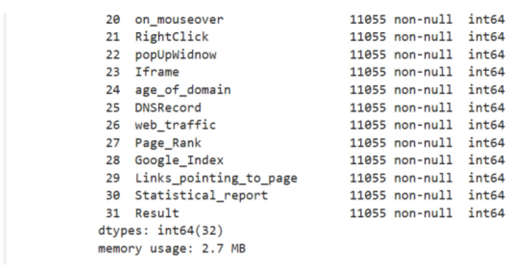
- Handling missing values
- Handling categorical data

Activity 5:Handling missing values

Lets find the shape of our dataset first. To find the shape of our data, the data. Shape method is used. To find the data type , the data.info() function is used.

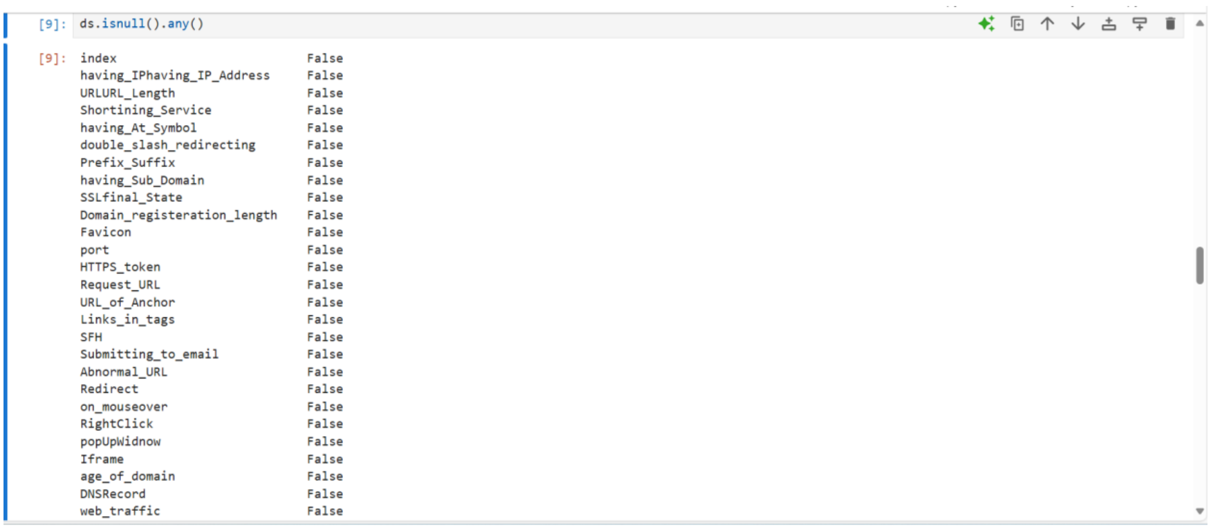


```
[7]: ds.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11055 entries, 0 to 11054
Data columns (total 32 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   index                       11055 non-null  int64
 1   having_IPhaving_IP_Address  11055 non-null  int64
 2   URLURL_Length               11055 non-null  int64
 3   Shortining_Service          11055 non-null  int64
 4   having_At_Symbol            11055 non-null  int64
 5   double_slash_redirecting    11055 non-null  int64
 6   Prefix_Suffix               11055 non-null  int64
 7   having_Sub_Domain           11055 non-null  int64
 8   SSLfinal_State              11055 non-null  int64
 9   Domain_registeration_length 11055 non-null  int64
 10  Favicon                     11055 non-null  int64
 11  port                        11055 non-null  int64
 12  HTTPS_token                 11055 non-null  int64
 13  Request_URL                 11055 non-null  int64
 14  URL_of_Anchor               11055 non-null  int64
 15  Links_in_tags               11055 non-null  int64
 16  SFH                         11055 non-null  int64
 17  Submitting_to_email         11055 non-null  int64
 18  Abnormal_URL                11055 non-null  int64
 19  Redirect                    11055 non-null  int64
 20  on_mouseover                11055 non-null  int64
 21  RightClick                  11055 non-null  int64
```

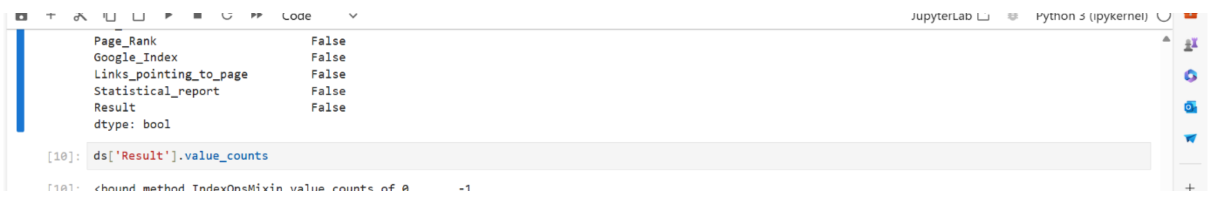


```
 20  on_mouseover                11055 non-null  int64
 21  RightClick                  11055 non-null  int64
 22  popUpWidnow                 11055 non-null  int64
 23  Iframe                      11055 non-null  int64
 24  age_of_domain               11055 non-null  int64
 25  DNSRecord                   11055 non-null  int64
 26  web_traffic                 11055 non-null  int64
 27  Page_Rank                   11055 non-null  int64
 28  Google_Index                11055 non-null  int64
 29  Links_pointing_to_page      11055 non-null  int64
 30  Statistical_report          11055 non-null  int64
 31  Result                      11055 non-null  int64
dtypes: int64(32)
memory usage: 2.7 MB
```

For checking the null values, data.isnull() function is used. From the below image, we found that there are no null values present in our dataset. So we can skip handling the missing values step.



```
[9]: ds.isnull().any()

[9]: index                           False
     having_IPhaving_IP_Address      False
     URLURL_Length                   False
     Shortining_Service              False
     having_At_Symbol                False
     double_slash_redirecting        False
     Prefix_Suffix                   False
     having_Sub_Domain               False
     SSLfinal_State                  False
     Domain_registeration_length     False
     Favicon                         False
     port                            False
     HTTPS_token                     False
     Request_URL                     False
     URL_of_Anchor                   False
     Links_in_tags                   False
     SFH                             False
     Submitting_to_email             False
     Abnormal_URL                    False
     Redirect                        False
     on_mouseover                    False
     RightClick                      False
     popUpWidnow                     False
     Iframe                          False
     age_of_domain                   False
     DNSRecord                       False
     web_traffic                     False
```

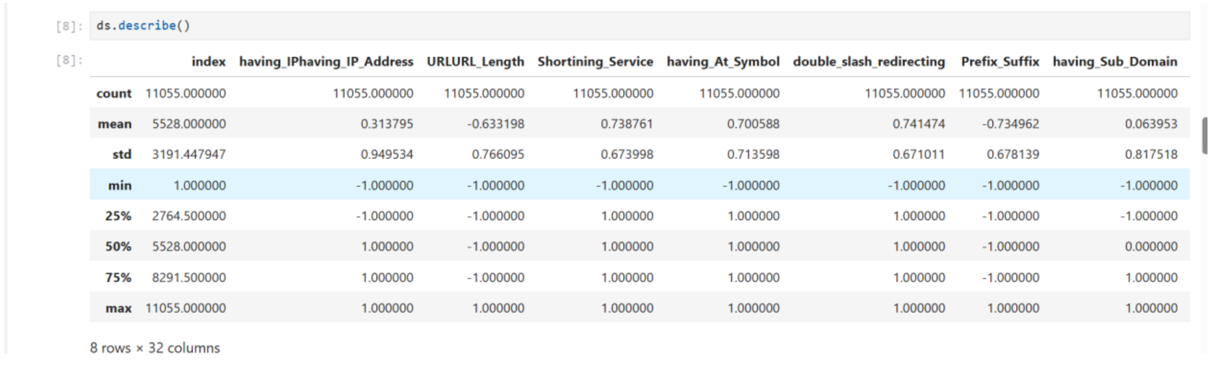


```
     Page_Rank                       False
     Google_Index                    False
     Links_pointing_to_page          False
     Statistical_report              False
     Result                          False
     dtype: bool

[10]: ds['Result'].value_counts

[10]: <bound method IndexOpsMixin.value_counts of 0        -1
```

Milestone 2: Exploratory Data Analysis

Activity 1: Descriptive statistics

Descriptive analysis is to study the basic feature of data with statistical process. Here pandas have a worthy function called describe. With this describe function we can understand the unique, top, and frequent values of categorical features. And we can find mean, std, min, max, and percentile values of continuous features.



```
[8]: ds.describe()
```

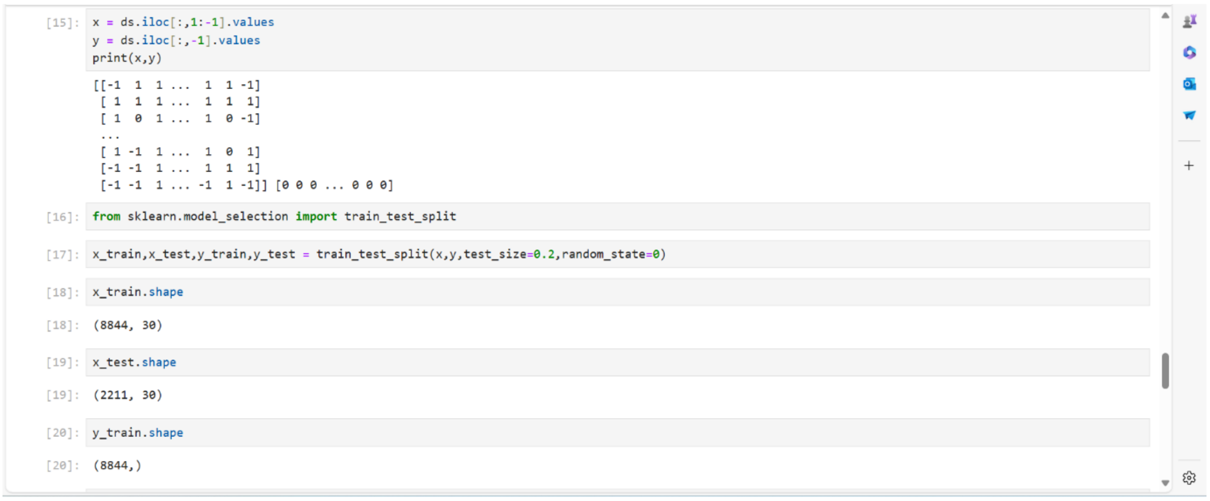
	index	having_IPhaving_IP_Address	URLURL_Length	Shortining_Service	having_At_Symbol	double_slash_redirecting	Prefix_Suffix	having_Sub_Domain
count	11055.000000	11055.000000	11055.000000	11055.000000	11055.000000	11055.000000	11055.000000	11055.000000
mean	5528.000000	0.313795	-0.633198	0.738761	0.700588	0.741474	-0.734962	0.063953
std	3191.447947	0.949534	0.766095	0.673998	0.713598	0.671011	0.678139	0.817518
min	1.000000	-1.000000	-1.000000	-1.000000	-1.000000	-1.000000	-1.000000	-1.000000
25%	2764.500000	-1.000000	-1.000000	1.000000	1.000000	1.000000	-1.000000	-1.000000
50%	5528.000000	1.000000	-1.000000	1.000000	1.000000	1.000000	-1.000000	0.000000
75%	8291.500000	1.000000	-1.000000	1.000000	1.000000	1.000000	-1.000000	1.000000
max	11055.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

8 rows × 32 columns

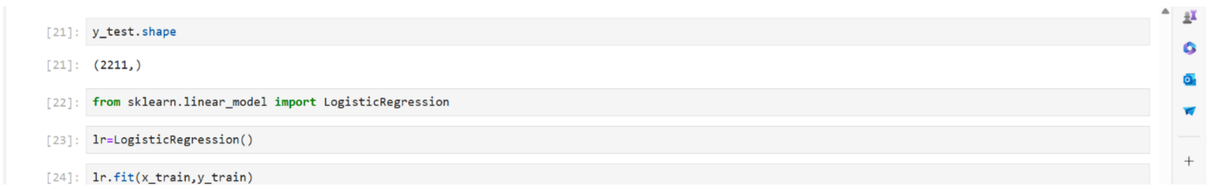
Activity 2: Splitting data into train and test

Now let’s split the dataset into train and test sets. First, split the dataset into x and y and then split the data set, here x and y variables are created. On the x variable, data is passed by dropping the target variable.

And on y target variable is passed. For splitting training and testing data we are using the train_test_split() function from sklearn. As parameters, we are passing x,y,test_size,random_state.



```
[15]: x = ds.iloc[:,1:-1].values
      y = ds.iloc[:,-1].values
      print(x,y)

      [[-1  1  1 ...  1  1 -1]
       [ 1  1  1 ...  1  1  1]
       [ 1  0  1 ...  1  0 -1]
       ...
       [ 1 -1  1 ...  1  0  1]
       [-1 -1  1 ...  1  1  1]
       [-1 -1  1 ... -1  1 -1]] [0 0 0 ... 0 0 0]

[16]: from sklearn.model_selection import train_test_split

[17]: x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2,random_state=0)

[18]: x_train.shape

[18]: (8844, 30)

[19]: x_test.shape

[19]: (2211, 30)

[20]: y_train.shape

[20]: (8844,)
```



```
[21]: y_test.shape

[21]: (2211,)

[22]: from sklearn.linear_model import LogisticRegression

[23]: lr=LogisticRegression()

[24]: lr.fit(x_train,y_train)
```

The code imports the MinMaxScaler module from sci-kit-learn for feature scaling. It then scales the feature in the input data ‘x’ using standardization and splits the dataset into training and testing sets with a test size of 20% and random state of 0.

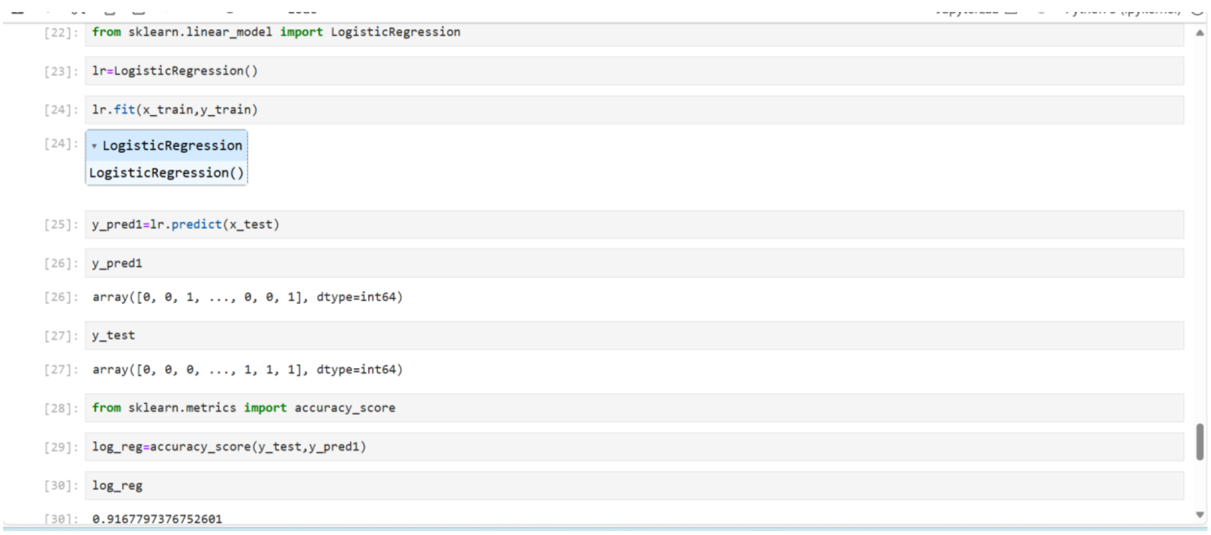
Milestone 3: Model Building

Activity 1: Training the model in multiple algorithms

Now our data is cleaned and it’s time to build the model. We can train our data on different algorithms. For this project, I am applying two classification algorithms. The best model is saved based on its performance.

Model 1:

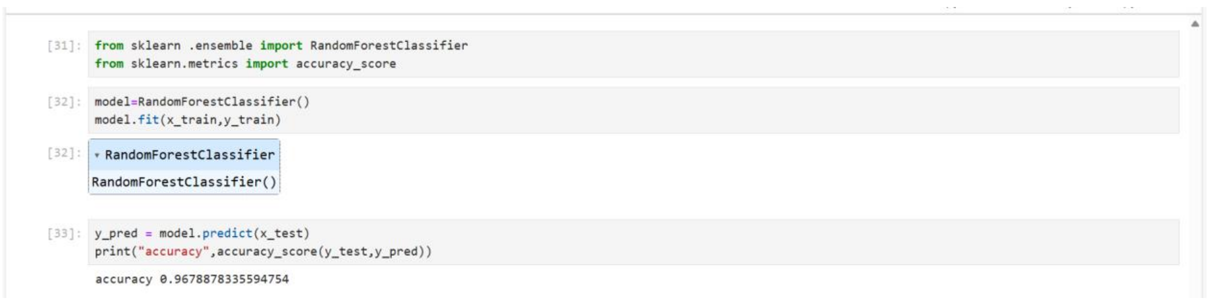
Logistic Regression



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

[23]: lr=LogisticRegression()

[24]: lr.fit(x_train,y_train)

[24]: ▾ LogisticRegression
      LogisticRegression()

[25]: y_pred1=lr.predict(x_test)

[26]: y_pred1

[26]: array([0, 0, 1, ..., 0, 0, 1], dtype=int64)

[27]: y_test

[27]: array([0, 0, 0, ..., 1, 1, 1], dtype=int64)

[28]: from sklearn.metrics import accuracy_score

[29]: log_reg=accuracy_score(y_test,y_pred1)

[30]: log_reg

[30]: 0.9167797376752601
```

Logistic regression is a data analysis technique that uses mathematics to find the relationships between two data factors. It then uses this relationship to predict the value of one of those factors based on the other. The prediction usually has a finite number of outcomes ,like phishing or safe.

Model 2: Random Forest



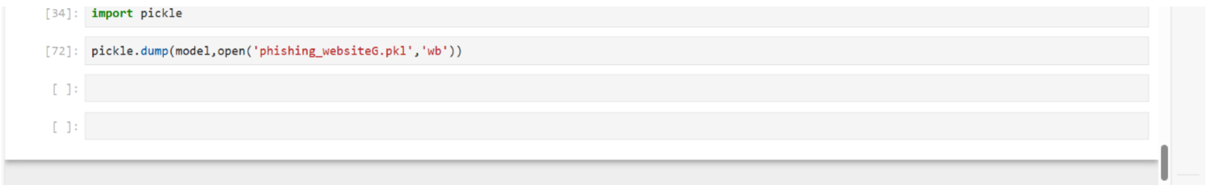
```
[31]: from sklearn .ensemble import RandomForestClassifier
      from sklearn.metrics import accuracy_score

[32]: model=RandomForestClassifier()
      model.fit(x_train,y_train)

[32]: ▾ RandomForestClassifier
      RandomForestClassifier()

[33]: y_pred = model.predict(x_test)
      print("accuracy",accuracy_score(y_test,y_pred))

      accuracy 0.9678878335594754
```

Random Forest is an ensemble learning method that constructs multiple decision trees during training and combines their predictions to improve accuracy and reduce overfitting. Each tree is trained on a random subset of the training data and a random subset of features. The final prediction is determined by averaging or taking a majority vote of the prediction made by individual tress. Random forest is widely used for classification and regression tasks and is known for its robustness and high performance on various types of datasets.

Milestone 4: Model Deployment

Activity 1: Save the best model

Saving the best model after comparing its performance using different evaluation metrics means selecting the model with the highest performance and saving its weights and configuration. This can be useful in avoiding the need to retrain the model every time it is needed and also to be able to use it in the future.



```
[34]: import pickle

[72]: pickle.dump(model,open('phishing_websiteG.pkl','wb'))

[ ]:

[ ]:
```

This code exports the trained machine learning model using pickle serialization and saves I as a file named “phishing_websiteG.pkl”. This allows for the model to be easily stored, shared, and loaded for future use without needing to retrain it.

Activity 2: Integrated with Web Framework

In this section, we will be building a web application that is integrated into the model we built. A UI is provided for the user where he has to enter the values for predictions. The enter values are given to the saved model and prediction is showcase on the UI.

This section has the following tasks

- Building HTML pages
- building server-side script
- run the web application
-

Activity 3: Building HTML pages:

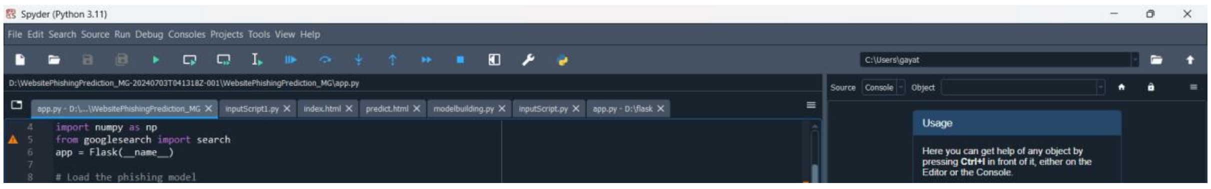
For this project create two HTML files namely

- index.html
- final.html

and save them in the templates folder.

Activity 4: Build python code:

Import the libraries



```
import numpy as np
from googlesearch import search
app = Flask(__name__)

# Load the phishing model
```

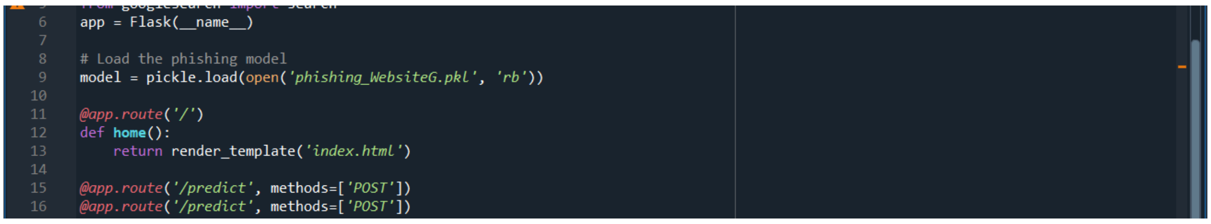
Load the saved model. importing the Flask module in the project is mandatory. An object of the Flask class is our WSGI application. The Flask constructor takes the name of the current module(__name__) as argument.

Here we will be using a declared constructor to route to HTML page that we have created earlier.

In the below image, the '/' URL is bound with the index.html function. Hence, when the index page of the web server is opened in the browser, HTML page will be rendered.

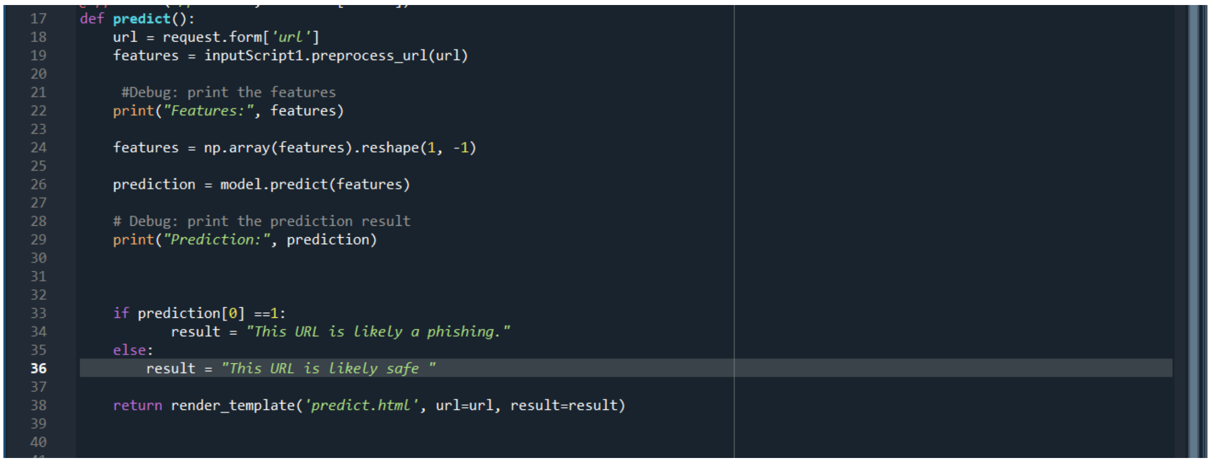
Whenever you enter the values from the html page the values can be retrieved using POST method.

Retrieve the values from UI:



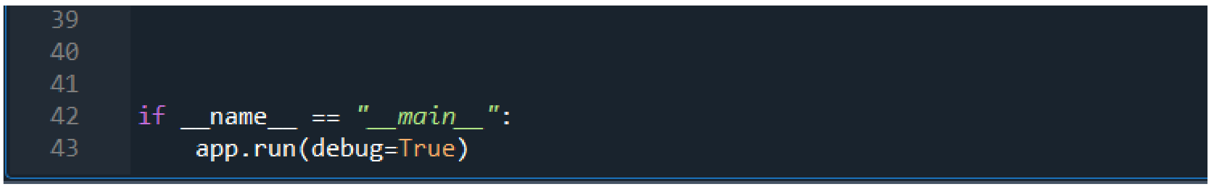
```
app = Flask(__name__)

# Load the phishing model
model = pickle.load(open('phishing_WebsiteG.pkl', 'rb'))

@app.route('/')
def home():
    return render_template('index.html')

@app.route('/predict', methods=['POST'])
@app.route('/predict', methods=['POST'])
```

Here we are routing our app to predict the function. This function retrieves all the values from the HTML pages using a post request. That is stored in an array. This array is passed to the model.predict() function. This function returns the prediction. This prediction value will be rendered to the text we have mentioned in the final.hml page earlier.



```
def predict():
    url = request.form['url']
    features = inputScript1.preprocess_url(url)

     #Debug: print the features
    print("Features:", features)

    features = np.array(features).reshape(1, -1)

    prediction = model.predict(features)

    # Debug: print the prediction result
    print("Prediction:", prediction)



    if prediction[0] ==1:
           result = "This URL is likely a phishing."
    else:
        result = "This URL is likely safe "

    return render_template('predict.html', url=url, result=result)
```

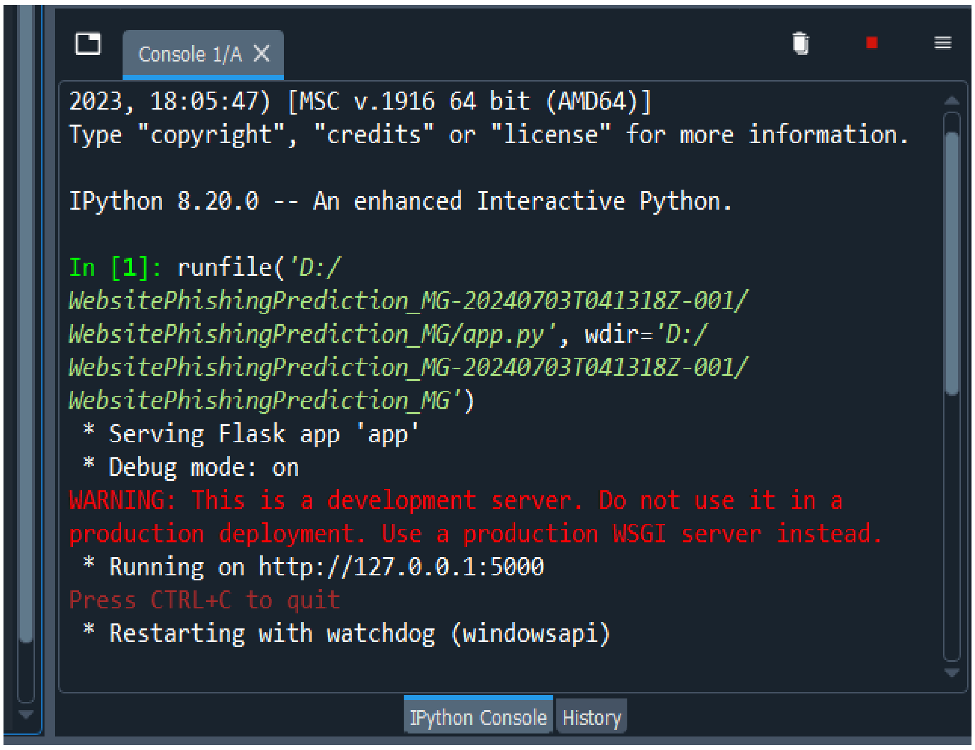
Main function:



```
if __name__ == "__main__":
    app.run(debug=True)
```

Activity 5: Run the web application

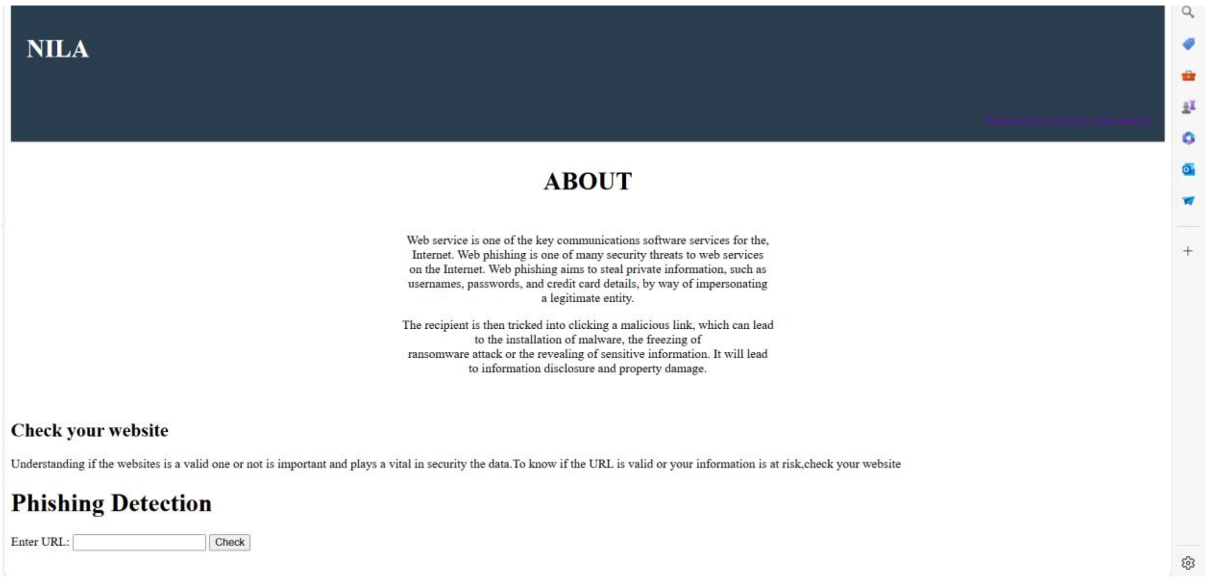
- open spyder
- navigate to the folder where python script is.
- Now open “app.py” file in spyder
- Navigate to the localhost where you can view your web page.
- Click on the predict button from down left corner after entering input, click predict button and see the result/prediction on the web page.



```
2023, 18:05:47) [MSC v.1916 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 8.20.0 -- An enhanced Interactive Python.

In [1]: runfile('D:/
WebsitePhishingPrediction_MG-20240703T041318Z-001/
WebsitePhishingPrediction_MG/app.py', wdir='D:/
WebsitePhishingPrediction_MG-20240703T041318Z-001/
WebsitePhishingPrediction_MG')
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a
production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with watchdog (windowsapi)
```

Now, go to the web browser and write the localhost URL(https://127.0.01:5000) to get the below result

RESULT:



NILA

ABOUT

Web service is one of the key communications software services for the, Internet. Web phishing is one of many security threats to web services on the Internet. Web phishing aims to steal private information, such as usernames, passwords, and credit card details, by way of impersonating a legitimate entity.

The recipient is then tricked into clicking a malicious link, which can lead to the installation of malware, the freezing of ransomware attack or the revealing of sensitive information. It will lead to information disclosure and property damage.

Check your website

Understanding if the websites is a valid one or not is important and plays a vital in security the data.To know if the URL is valid or your information is at risk,check your website

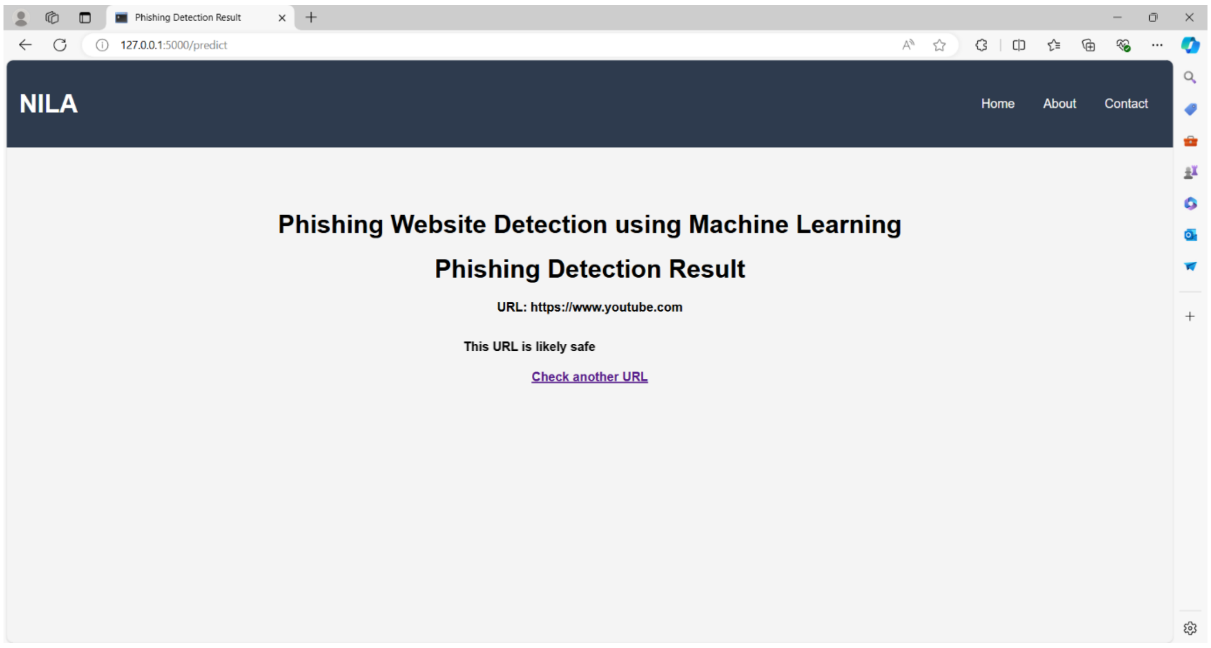
Phishing Detection

Enter URL: Check

in this page we give the respected value for the respected field.

Output:

In this final page the output that is phishing website or safe website will be shown.



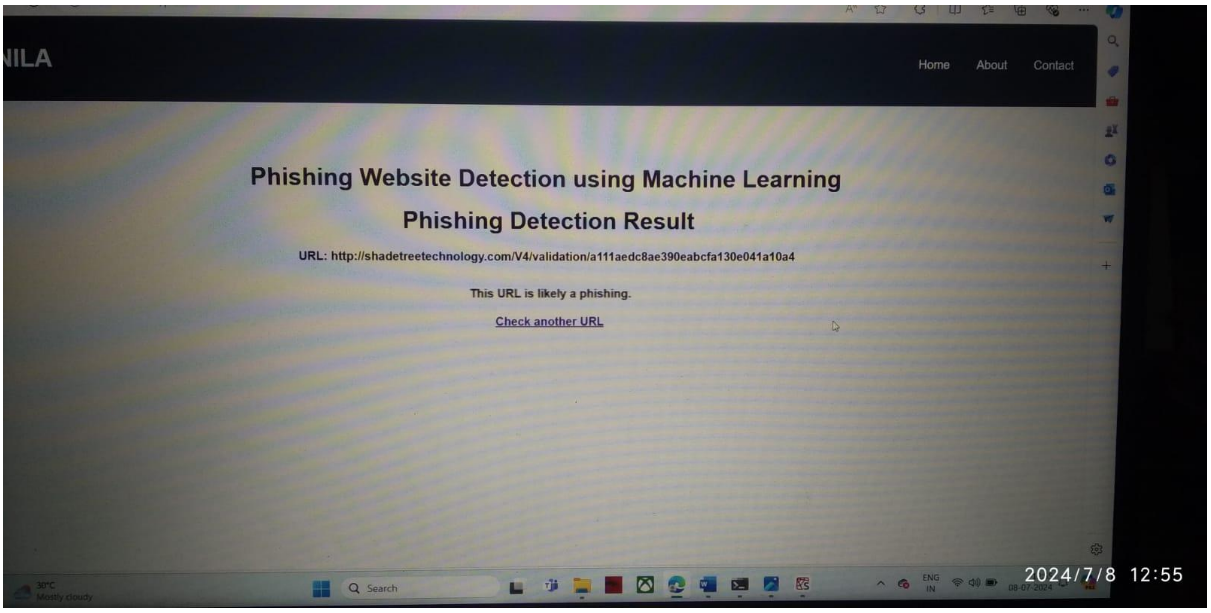
Phishing Website Detection using Machine Learning

Phishing Detection Result

URL: https://www.youtube.com

This URL is likely safe

Check another URL



Phishing Website Detection using Machine Learning

Phishing Detection Result

URL: http://shadetreetechnology.com/V4/validation/a111aedc8ae390eabcfa130e041a10a4

This URL is likely a phishing.

Check another URL

5.CONCLUSION

In conclusion, phishing website detection using machine learning technique is a promising and effective approach to identify potentially malicious websites. The utilization of machine learning algorithms enables automated and scalable detection, offering advantages over traditional rule-based or manual methods. Machine learning techniques such as Random Forest and logistic regression demonstrate effectiveness in distinguishing between phishing and legitimate websites. These algorithms can capture complex patterns and relationships within the data. Security is a critical aspect, and the model's deployment.

Phishing website detection using machine learning is adynamic and evolving field. A successful implementation requires a thoughtful approach to data collection, model training, evaluation and deployment. Regular monitoring and adaptation to emerging threats contribute to the sustained effectiveness of the detection system.

5.1 FUTURE SCOPE

The feature scope for phishing website detection using machine learning techniques encompasses a wide range of characteristics that can help distinguish between legitimate and malicious websites. These features are designed to capture various aspects of the website's structure, content, behaviour and other relevant attributes. Here is a comprehensive breakdown of the feature scope

This feature scope is extensive and covers various aspects of website that can be indicative of phishing behaviour. The choice of features may depend on the specific characteristics of the dataset, the machine learning algorithms being used, and the evolving nature of phishing techniques. Regular updates and adaptations to the feature scope are crucial for maintaining the effectiveness of phishing detection models.

6.BILIOGRAPHY

- https://www.kaggle.com/datasets/shashwatwork/web-page-phshing-detection-dataset