

**URL-Based Phishing Detection Using Machine Learning**



**Team Members**

B.Sumanjali(20481A1221) (Team Lead)

A. Jahnavi (20481A1205)

Ch. Sushma (20481A1227)

D. Srivalli (20481A1237)

**Table of Contents**

**Topic Page No.**

1. Introduction **1** 
   1. Overview **1**
   2. Purpose **1**
2. Literature Survey **2** 
   1. Existing problem **2**
   2. Proposed solution **2**
3. Theoritical Analysis **3** 
   1. Block diagram **3**
   2. Software designing **3**
4. Experimental Investigations **4-18**
5. Flowchart 1**9**
6. Result **20-21**
7. Advantages & Disadvantages **22**
8. Applications **23**
9. Conclusion **23**
10. Future Scope **24**
11. Biblography **24** 
    1. Source Code **25-26**

## 1. INTRODUCTION

#### Overview: -

There are a number of users who purchase products online and make payments through e-banking. There are e-banking websites that ask users to provide sensitive data such as username, password & credit card details, etc., often for malicious reasons. This type of e-banking website is known as a phishing website. 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.

Common threats of web phishing:

* Web phishing aims to steal private information, such as usernames, passwords, and credit card details, by way of impersonating a legitimate entity.
* It will lead to information disclosure and property damage.
* Large organizations may get trapped in different kinds of scams.

#### 1.2 Purpose: -

This Guided Project mainly focuses on applying a machine-learning algorithm to detect Phishing websites.In order to detect and predict e-banking phishing websites, we proposed an intelligent, flexible and effective system that is based on using classification algorithms.  We implemented classification algorithms and techniques to extract the phishing datasets criteria to classify their legitimacy. The e-banking phishing website can be detected based on some important characteristics like URL and domain identity, and security and encryption criteria in the final phishing detection rate. Once a user makes a transaction online when he makes payment through an e-banking website our system will use a data mining algorithm to detect whether the e-banking website is a phishing website or not.

# 2. LITERATURE SURVEY

#### 2.1 Existing problem: -

Imbalanced Datasets: The dataset used for training ML models may be imbalanced, meaning there might be significantly more legitimate URLs than phishing URLs, affecting the model's ability to accurately detect phishing attempts.

Data Privacy Concerns: The analysis of URLs may inadvertently reveal sensitive information or violate user privacy, raising ethical and legal concerns.

Scalability: As the volume of URLs increases, the computational resources and time required for processing and analysis can become a challenge.

#### 2.2 Proposed solution: -

Feature Engineering: Extract relevant features from URLs, such as domain reputation, URL length, presence of certain keywords, and domain age. These features can provide valuable information for ML models to distinguish between legitimate and phishing URLs.

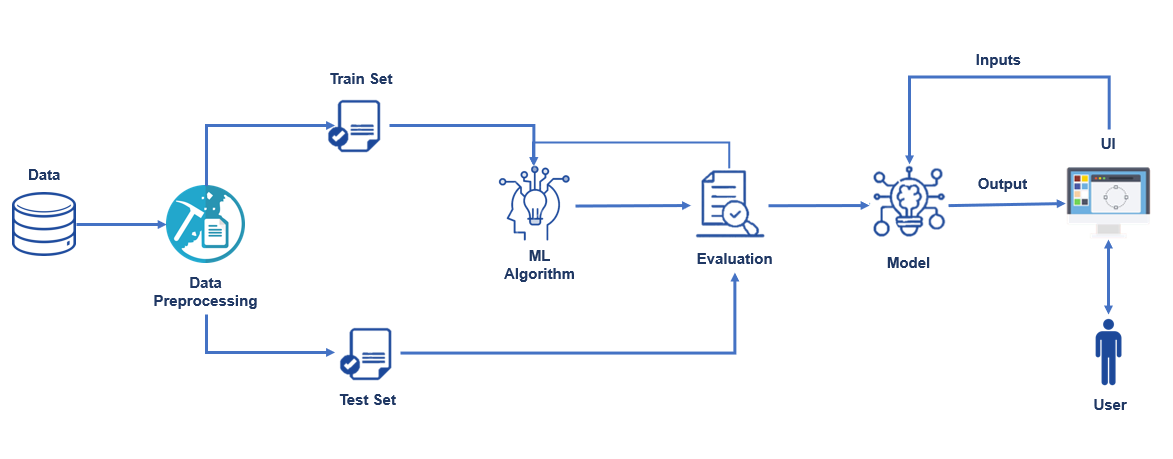
Ensemble Learning: Combine multiple ML models, such as Random Forests, Gradient Boosting, and Decision tree classification in an ensemble to leverage their strengths and mitigate individual weaknesses, improving overall detection performance.

Imbalanced Data Handling: Address imbalanced datasets by using techniques like oversampling, undersampling, or using synthetic data generation methods like SMOTE (Synthetic Minority Over-sampling Technique) to balance the proportion of legitimate and phishing URLs in the training data.

Transfer Learning: Utilize pre-trained models on large-scale datasets related to web data or URLs and fine-tune them with domain-specific phishing data to benefit from the knowledge and features learned from broader contexts.

**3. THEORETICAL ANALYSIS**

**3.1 Block diagram**



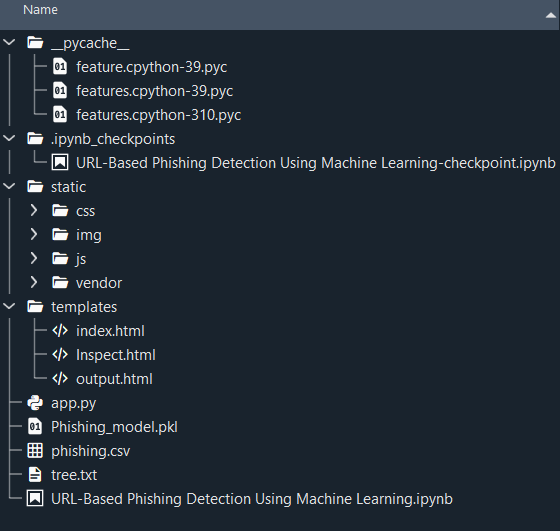
3.2 Software designing: -

To complete this project, you must required following software’s, concepts and packages

* **Anaconda navigator:**
  + Refer the link below to download anaconda navigator.
  + Link : <https://youtu.be/1ra4zH2G4o0>
* **Python packages:**
  + Open anaconda prompt as administrator
  + Type “pip install numpy” and click enter.
  + Type “pip install pandas” and click enter.
  + Type “pip install scikit-learn” and click enter.
  + Type ”pip install matplotlib” and click enter.
  + Type ”pip install scipy” and click enter.
  + Type ”pip install pickle-mixin” and click enter.
  + Type ”pip install seaborn” and click enter.
  + Type “pip install Flask” and click enter.

# 4. EXPERIMENTAL INVESTIGATIONS

Create the Project folder which contains files as shown below

****

### We are building a flask application which needs HTML pages stored in the templates folder.

# Milestone 1: Data Collection & Data Pre-processing

### Activity 1: Importing Required Libraries:

### 

### Collection Of Dataset

### To start with, we have to select or identify a dataset that contains a set of features through which a phishing website can be identified.

**Activity 2: Download the dataset**

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

In this project we have used phishing.csv data. This data is downloaded from kaggle.com. Please refer the link given below to download the dataset.

Dataset Link: https://www.kaggle.com/eswarchandt/phishing-website-detector

As we have understood how the data is collected lets pre-process the collected data.

**Activity 3: Data Pre-processing**

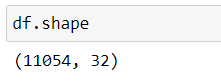
The download data set is not suitable for training the machine learning model as it might have so much of randomness so we need to clean the dataset properly in order to fetch good results. This activity includes the following steps.

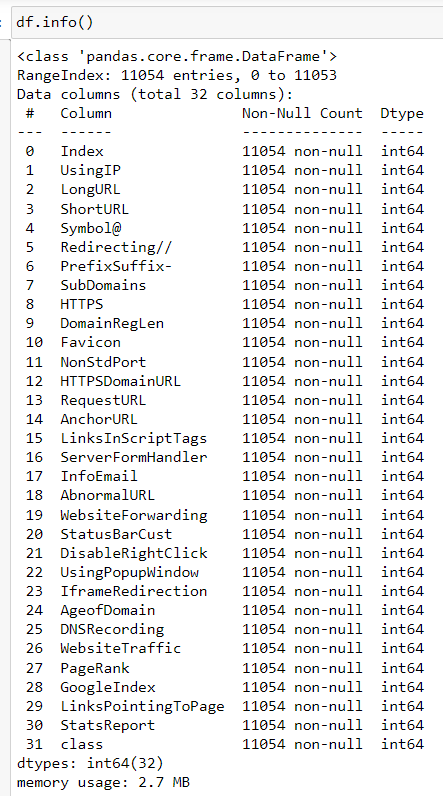
* Handling missing values
* Handling categorical data
* Handling outliers
* Scaling Techniques
* Splitting dataset into training and test set

Note: These are the general steps of pre-processing the data before using it for machine learning. Depending on the condition of your dataset, you may or may not have to go through all these steps.

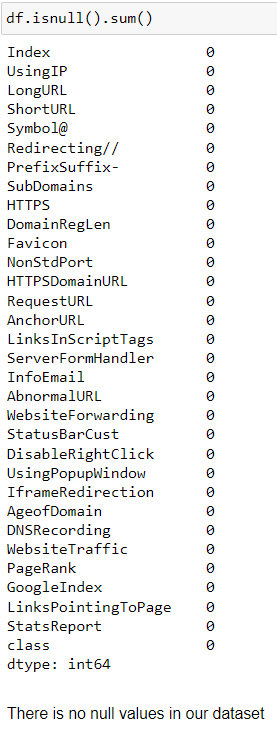
**Activity 4: Checking for null values**

Let’s find the shape of our dataset first, To find the shape of our data, df.shape method is used. To find the data type, df.info() function is used.

****

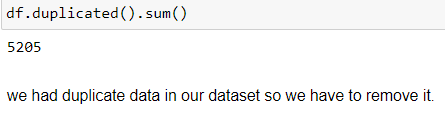
****

For checking the null values, df.isnull() function is used. To sum those null values we use .sum() function to it. From the below image we found that there are some null values present in our dataset. So we have to handle the missing values.

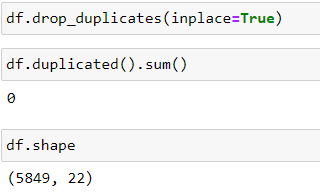


There is no categorical data in our dataset.

**Activiy 5: Checking for duplicated data**

****

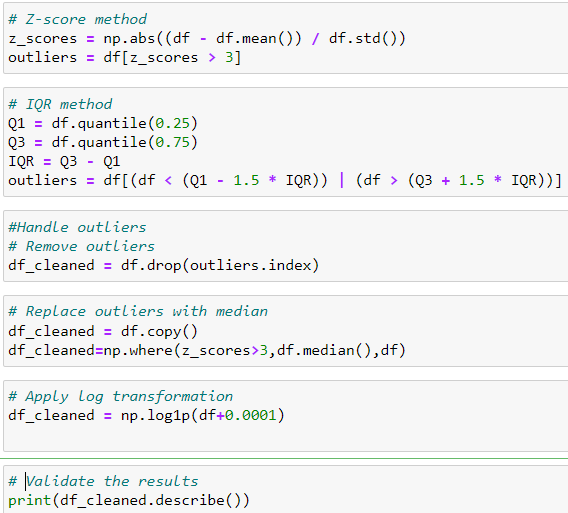
**Handling duplicate data**

****

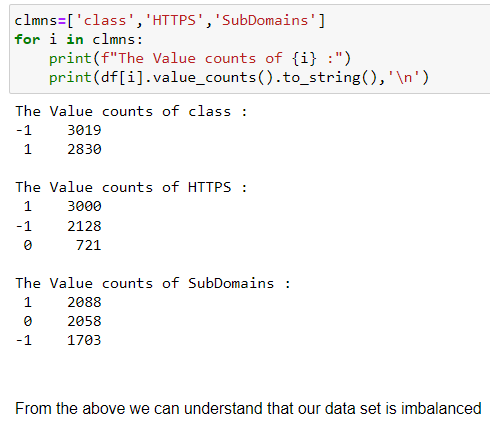
**Activity 6: Outliers**

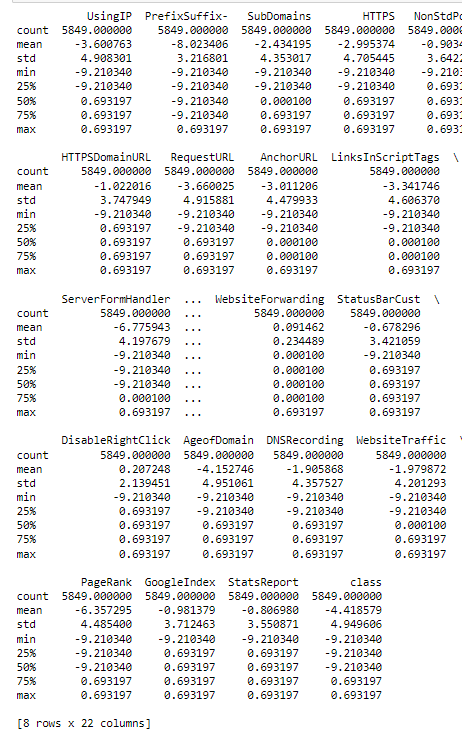
We had outliers in our dataset in the columns 'PrefixSuffix-','NonStdPort', 'HTTPSDomainURL','AnchorURL','ServerFormHandler','InfoEmail','AbnormalURL','WebsiteForwarding','StatusBarCust','DisableRightClick','GoogleIndex','StatsReport'

**Handling Outliers**

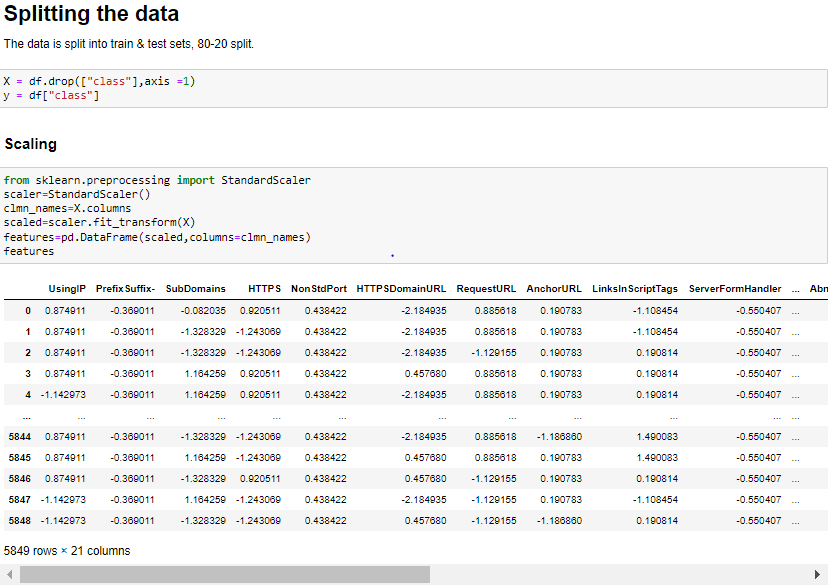
****

**Activity 7: Checking data is balanced or not?**

****

****

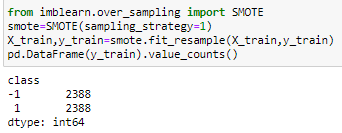
**Activity 8: Scaling**

****

### Activity 9: Train Test and Split

### 

**Activity 10: Handling Balance Data**

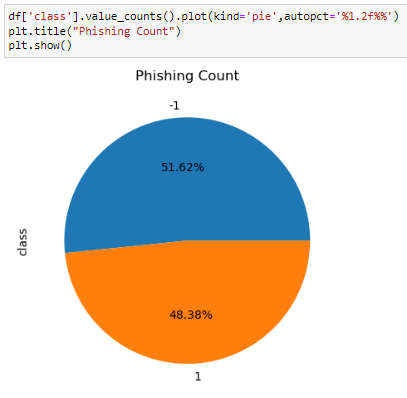
****

From the above we understand that our data is balanced.

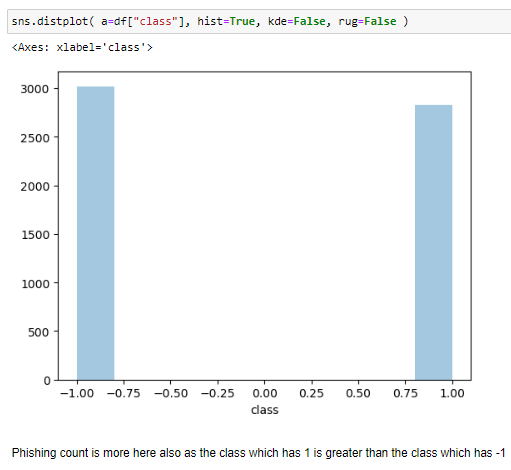
# Milestone 2: Visualizing and analysing the data

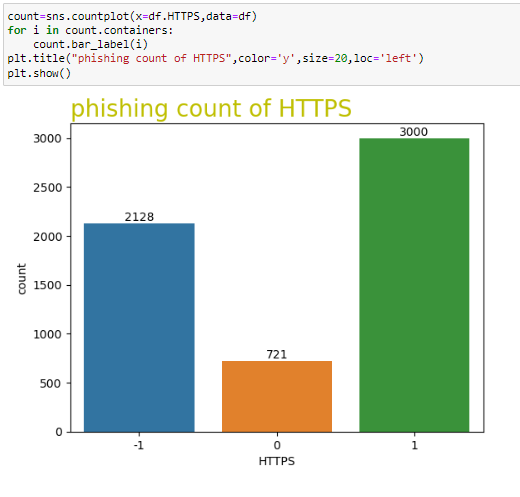
**Activity 1: Univariate analysis**

In simple words, univariate analysis is understanding the data with single feature. Here we have displayed two different graphs such as distplot and countplot.

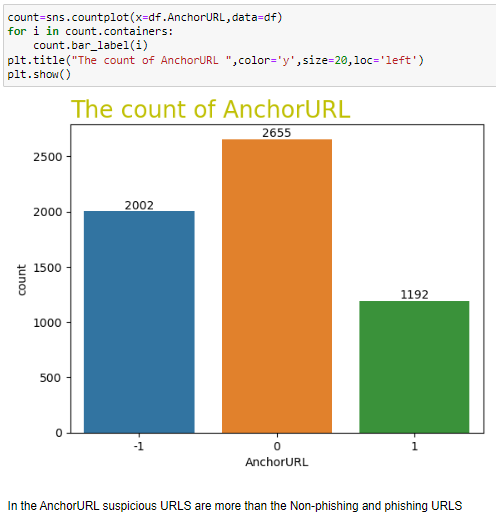
****

* From the above plot we came to know, the highest distribution of phishing is unsafe with 51.62%



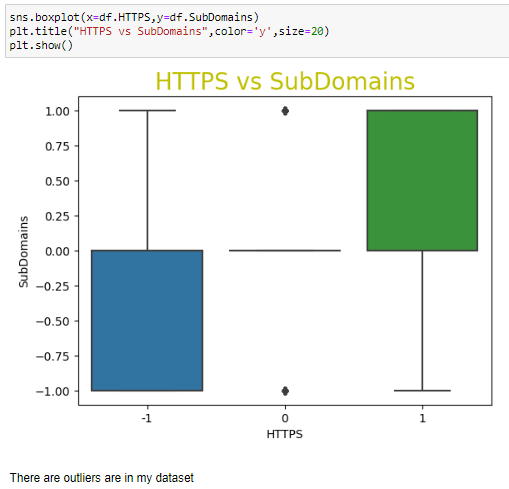


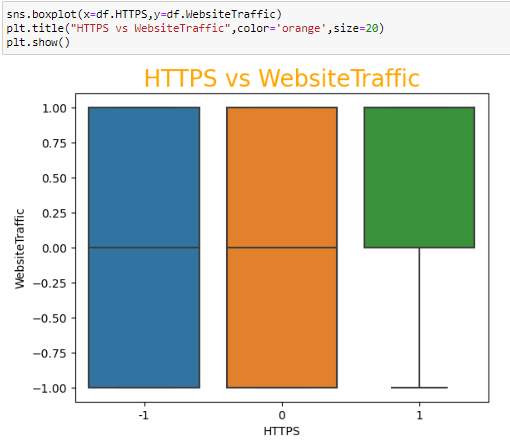
* phishing URLS are 2128.
* legitimate HTTPS URLS or non\_phishing URLS are 3000.
* Suspicious URLS are 721 (0 indicates a potential risk of features that indicate a potential risk of phishing, but they are not confirmed phishing URLS).

****

**Activity 2: Bivariate analysis**

To find the relation between two features we use bivariate analysis.

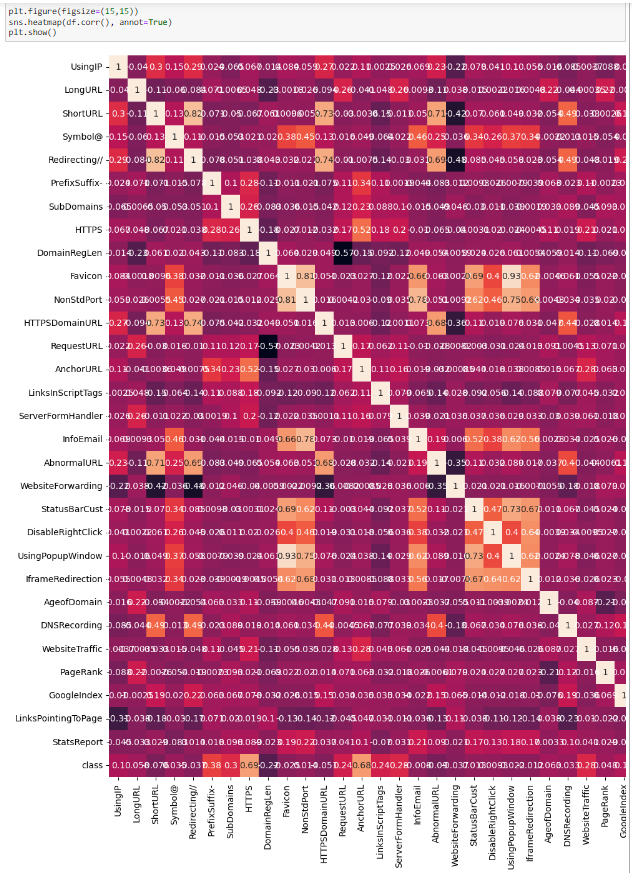
****

****

**Activity 3: Multivariate analysis**

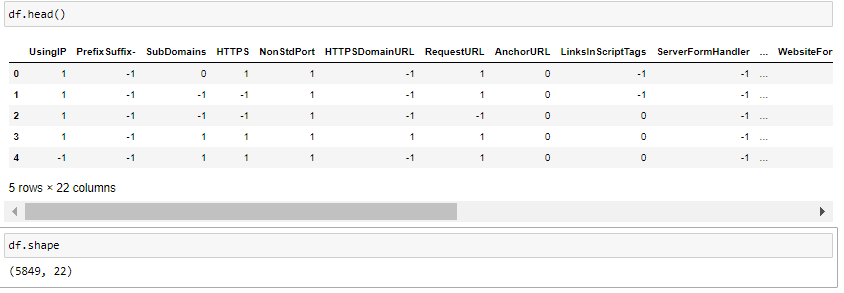
In simple words, multivariate analysis is to find the relation between multiple features. Here we have used heatmap from seaborn package.

* From the below image, we came to a conclusion that how data is distributed and how they are and how much they are correlated each other.
* All the features weather following the normal distribution or not ?

****

here in this dataset some features have no good reationship so we can delete those we can delete the columns based on dependent variable class So, here I'm going to delete LongURL,ShortURL,Symbol@, Redirecting //,DomainRegLen, Favicon, UsingPopupWindow, IframeRedirection, LinksPointingToPage





After completion of training and splitting the data we had 21 column.

* **Milestone 3**
* **Model Building and Comparision of Models**

There are two major types of supervised machine learning problems, called classification and regression. Our data set comes under regression problem, as the prediction of suicide rate is a continuous number, or a floating-point number in programming terms. The supervised machine learning models (regression) considered to train the dataset in this notebook are:Logistic Regression, K-Nearest Neighbors , Naive Bayes, Decision Tree, Random Forest, Gradient Boosting, Multi Layer perceptron Classifier, Support Vector Machine Classifier.

The metrics considered to evaluate the model performance are Accuracy & F1 score.

To compare the models performance, a dataframe is created. The columns of this dataframe are the lists created to store the results of the model.



Model: Logistic Regression

Accuracy: 0.9222222222222223

Confusion Matrix:

[[573 58]

[ 33 506]]

Classification Report:

precision recall f1-score support

-1 0.95 0.91 0.93 631

1 0.90 0.94 0.92 539

accuracy 0.92 1170

macro avg 0.92 0.92 0.92 1170

weighted avg 0.92 0.92 0.92 1170

--------------------------------------------------

Model: K-Nearest Neighbors

Accuracy: 0.9222222222222223

Confusion Matrix:

[[588 43]

[ 48 491]]

Classification Report:

precision recall f1-score support

-1 0.92 0.93 0.93 631

1 0.92 0.91 0.92 539

accuracy 0.92 1170

macro avg 0.92 0.92 0.92 1170

weighted avg 0.92 0.92 0.92 1170

--------------------------------------------------

Model: Naive Bayes

Accuracy: 0.6623931623931624

Confusion Matrix:

[[631 0]

[395 144]]

Classification Report:

precision recall f1-score support

-1 0.62 1.00 0.76 631

1 1.00 0.27 0.42 539

accuracy 0.66 1170

macro avg 0.81 0.63 0.59 1170

weighted avg 0.79 0.66 0.61 1170

--------------------------------------------------

Model: Decision Tree

Accuracy: 0.9145299145299145

Confusion Matrix:

[[591 40]

[ 60 479]]

Classification Report:

precision recall f1-score support

-1 0.91 0.94 0.92 631

1 0.92 0.89 0.91 539

accuracy 0.91 1170

macro avg 0.92 0.91 0.91 1170

weighted avg 0.91 0.91 0.91 1170

--------------------------------------------------

Model: Random Forest

Accuracy: 0.9282051282051282

Confusion Matrix:

[[586 45]

[ 39 500]]

Classification Report:

precision recall f1-score support

-1 0.94 0.93 0.93 631

1 0.92 0.93 0.92 539

accuracy 0.93 1170

macro avg 0.93 0.93 0.93 1170

weighted avg 0.93 0.93 0.93 1170

--------------------------------------------------

Model: Gradient Boosting

Accuracy: 0.9452991452991453

Confusion Matrix:

[[596 35]

[ 29 510]]

Classification Report:

precision recall f1-score support

-1 0.95 0.94 0.95 631

1 0.94 0.95 0.94 539

accuracy 0.95 1170

macro avg 0.94 0.95 0.95 1170

weighted avg 0.95 0.95 0.95 1170

--------------------------------------------------

Model: Multi-Layer Perceptron

Accuracy: 0.9401709401709402

Confusion Matrix:

[[587 44]

[ 26 513]]

Classification Report:

precision recall f1-score support

-1 0.96 0.93 0.94 631

1 0.92 0.95 0.94 539

accuracy 0.94 1170

macro avg 0.94 0.94 0.94 1170

weighted avg 0.94 0.94 0.94 1170

--------------------------------------------------

Model: Support Vector

Accuracy: 0.9384615384615385

Confusion Matrix:

[[584 47]

[ 25 514]]

Classification Report:

precision recall f1-score support

-1 0.96 0.93 0.94 631

1 0.92 0.95 0.93 539

accuracy 0.94 1170

macro avg 0.94 0.94 0.94 1170

weighted avg 0.94 0.94 0.94 1170

--------------------------------------------------



Model: Gradient Boosting

Accuracy: 0.9452991452991453

Confusion Matrix:

[[596 35]

[ 29 510]]

Classification Report:

precision recall f1-score support

-1 0.95 0.94 0.95 631

1 0.94 0.95 0.94 539

accuracy 0.95 1170

macro avg 0.94 0.95 0.95 1170

weighted avg 0.95 0.95 0.95 1170

--------------------------------------------------

Model: Random Forest

Accuracy: 0.941025641025641

Confusion Matrix:

[[596 35]

[ 34 505]]

Classification Report:

precision recall f1-score support

-1 0.95 0.94 0.95 631

1 0.94 0.94 0.94 539

accuracy 0.94 1170

macro avg 0.94 0.94 0.94 1170

weighted avg 0.94 0.94 0.94 1170

--------------------------------------------------

Model: Multi-Layer Perceptron

Accuracy: 0.9393162393162393

Confusion Matrix:

[[594 37]

[ 34 505]]

Classification Report:

precision recall f1-score support

-1 0.95 0.94 0.94 631

1 0.93 0.94 0.93 539

accuracy 0.94 1170

macro avg 0.94 0.94 0.94 1170

weighted avg 0.94 0.94 0.94 1170

--------------------------------------------------

Model: Support Vector

Accuracy: 0.9384615384615385

Confusion Matrix:

[[584 47]

[ 25 514]]

Classification Report:

precision recall f1-score support

-1 0.96 0.93 0.94 631

1 0.92 0.95 0.93 539

accuracy 0.94 1170

macro avg 0.94 0.94 0.94 1170

weighted avg 0.94 0.94 0.94 1170

--------------------------------------------------

Model: Logistic Regression

Accuracy: 0.9222222222222223

Confusion Matrix:

[[573 58]

[ 33 506]]

Classification Report:

precision recall f1-score support

-1 0.95 0.91 0.93 631

1 0.90 0.94 0.92 539

accuracy 0.92 1170

macro avg 0.92 0.92 0.92 1170

weighted avg 0.92 0.92 0.92 1170

--------------------------------------------------

Model: K-Nearest Neighbors

Accuracy: 0.9222222222222223

Confusion Matrix:

[[588 43]

[ 48 491]]

Classification Report:

precision recall f1-score support

-1 0.92 0.93 0.93 631

1 0.92 0.91 0.92 539

accuracy 0.92 1170

macro avg 0.92 0.92 0.92 1170

weighted avg 0.92 0.92 0.92 1170

--------------------------------------------------

Model: Decision Tree

Accuracy: 0.9128205128205128

Confusion Matrix:

[[589 42]

[ 60 479]]

Classification Report:

precision recall f1-score support

-1 0.91 0.93 0.92 631

1 0.92 0.89 0.90 539

accuracy 0.91 1170

macro avg 0.91 0.91 0.91 1170

weighted avg 0.91 0.91 0.91 1170

--------------------------------------------------

Model: Naive Bayes

Accuracy: 0.6623931623931624

Confusion Matrix:

[[631 0]

[395 144]]

Classification Report:

precision recall f1-score support

-1 0.62 1.00 0.76 631

1 1.00 0.27 0.42 539

accuracy 0.66 1170

macro avg 0.81 0.63 0.59 1170

weighted avg 0.79 0.66 0.61 1170

-----------------------------------------------

## 5. FLOWCHART

## 

## 

### 6. RESULT

For this project create three HTML files namely

* index.html
* inspect.html
* output.html

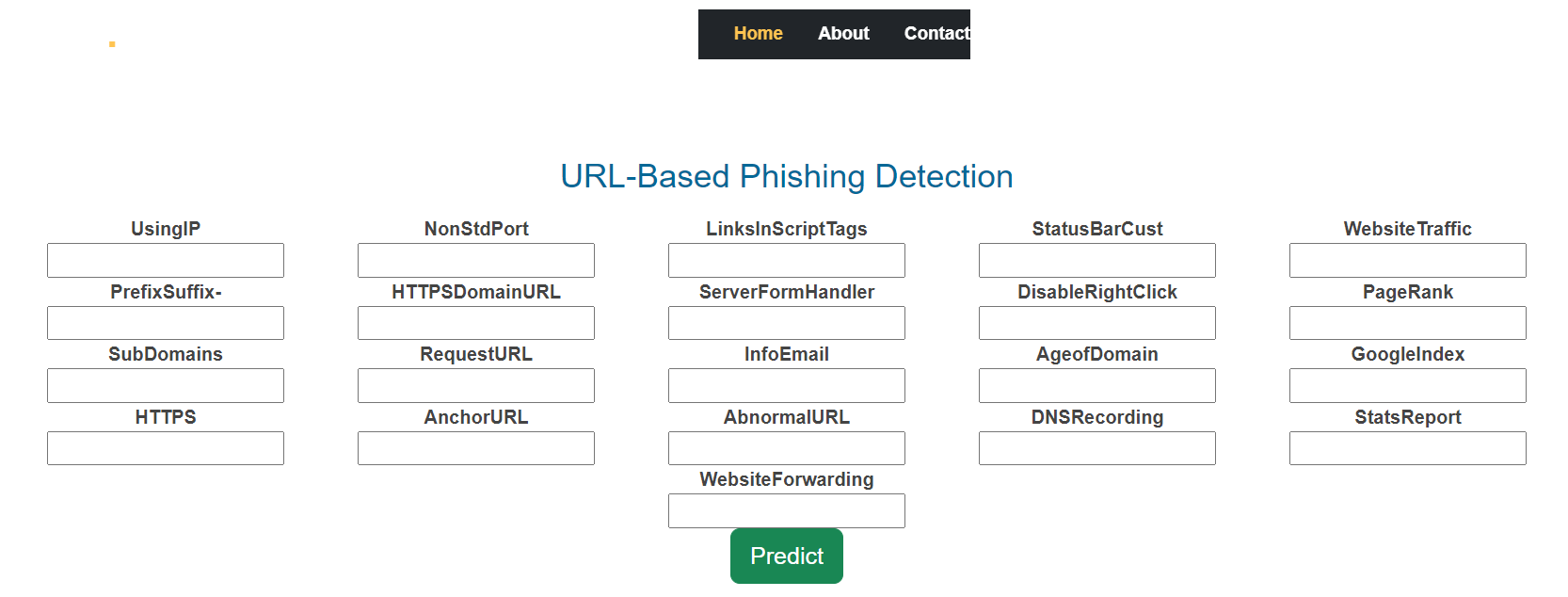
and save them in templates folder.

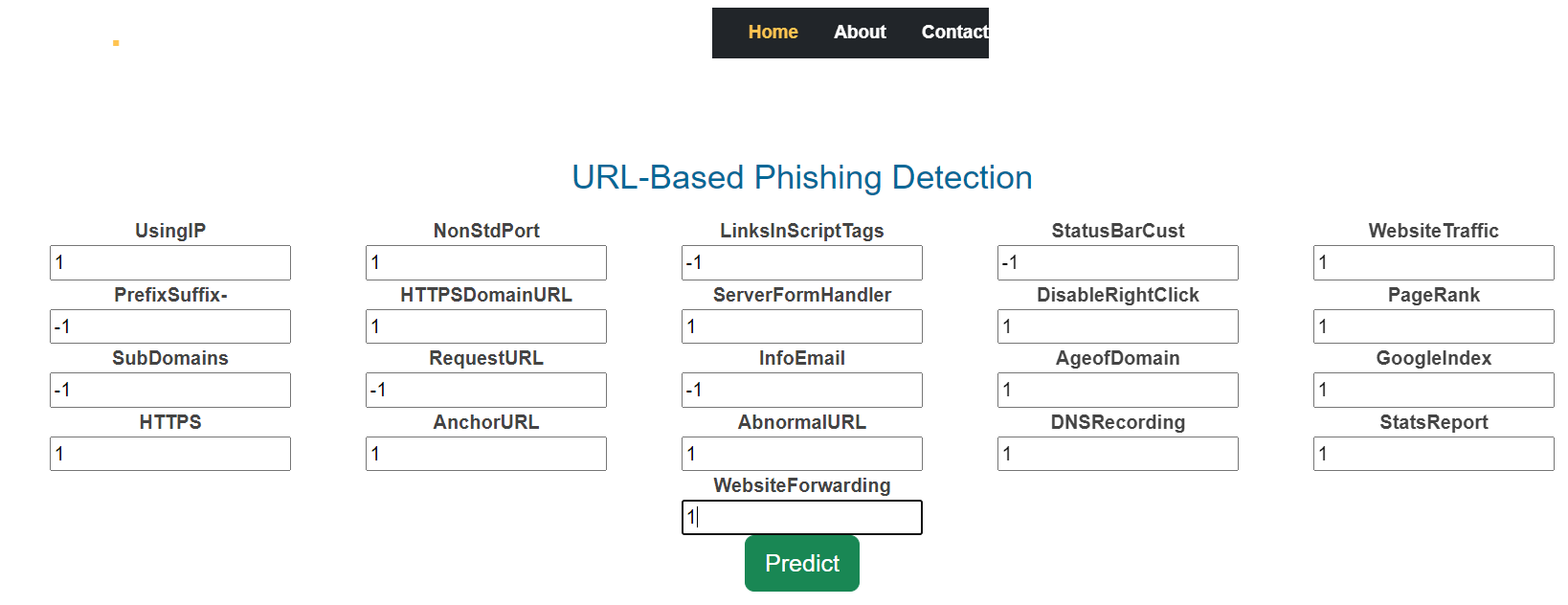
This is how our index.html page looks like:

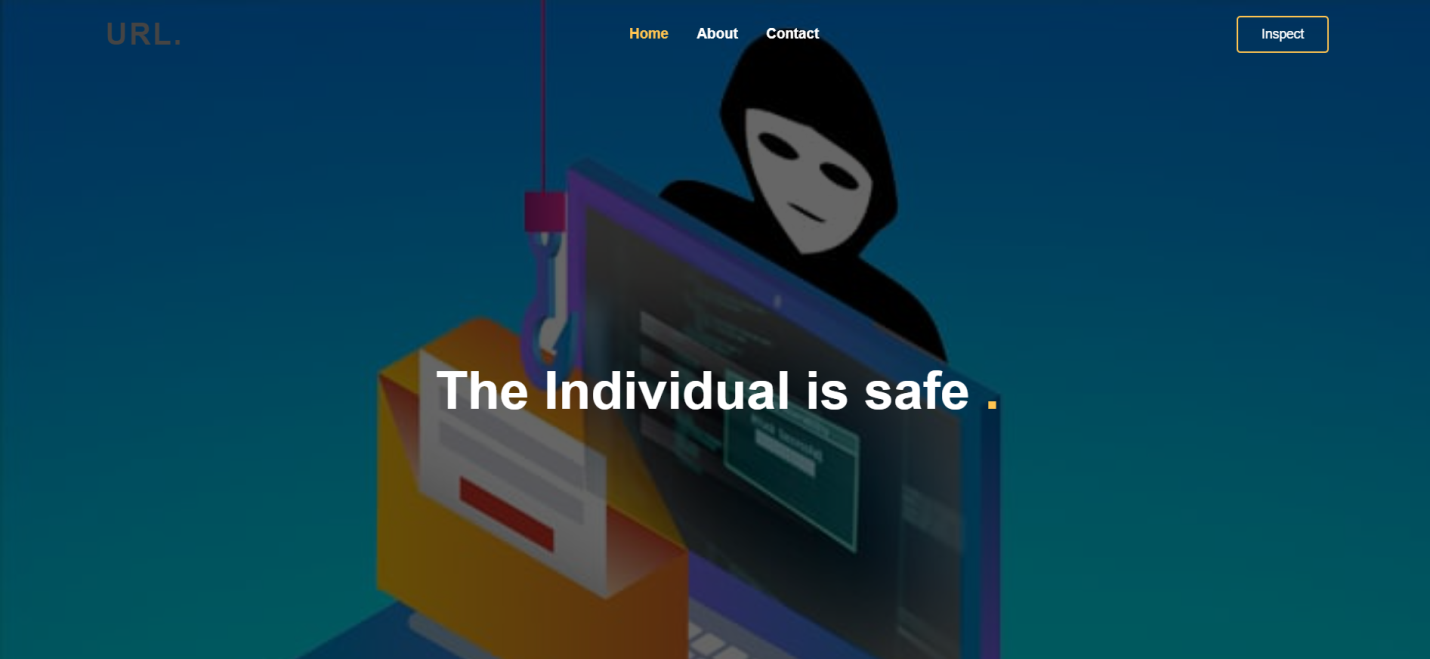


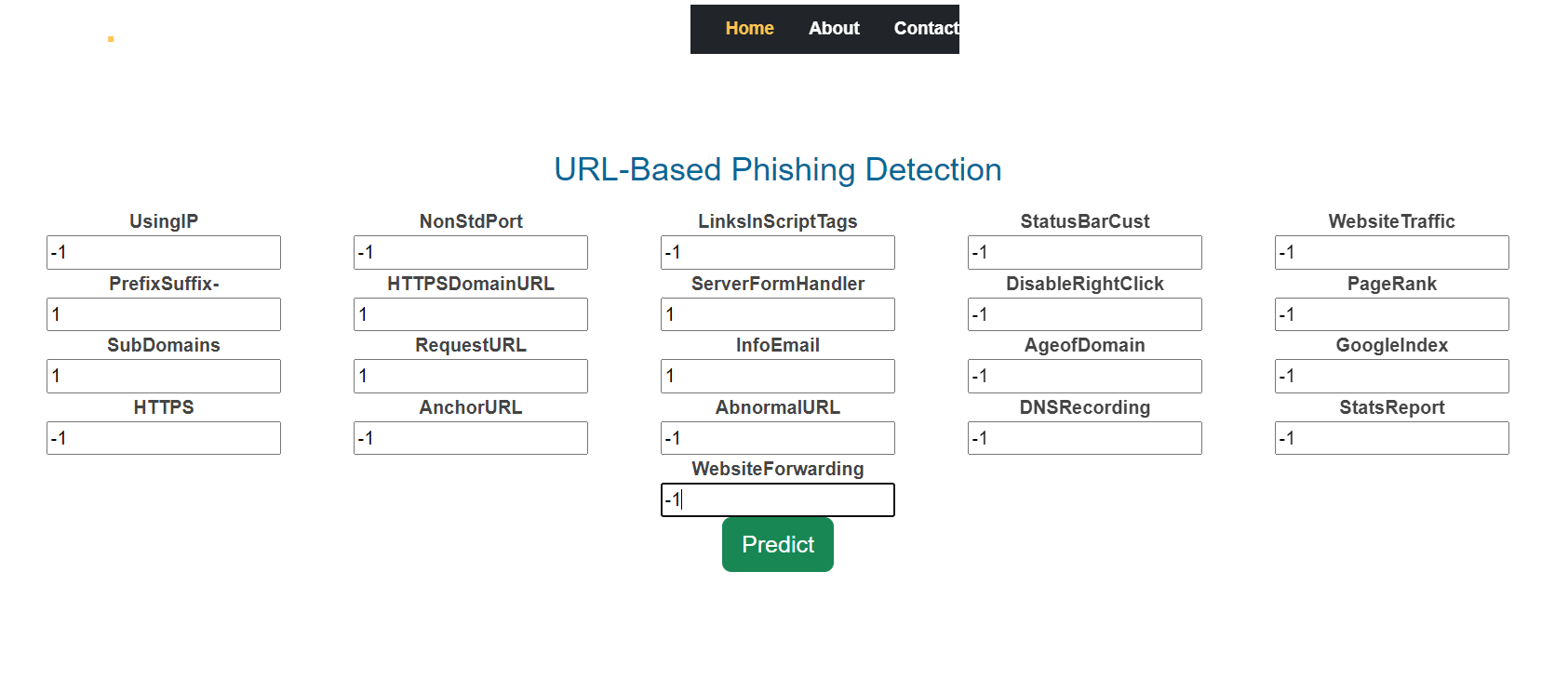
Now when you click on inspect button from top right corner you will get redirected to Inspect.html

Lets look how our Inspect.html file looks like:

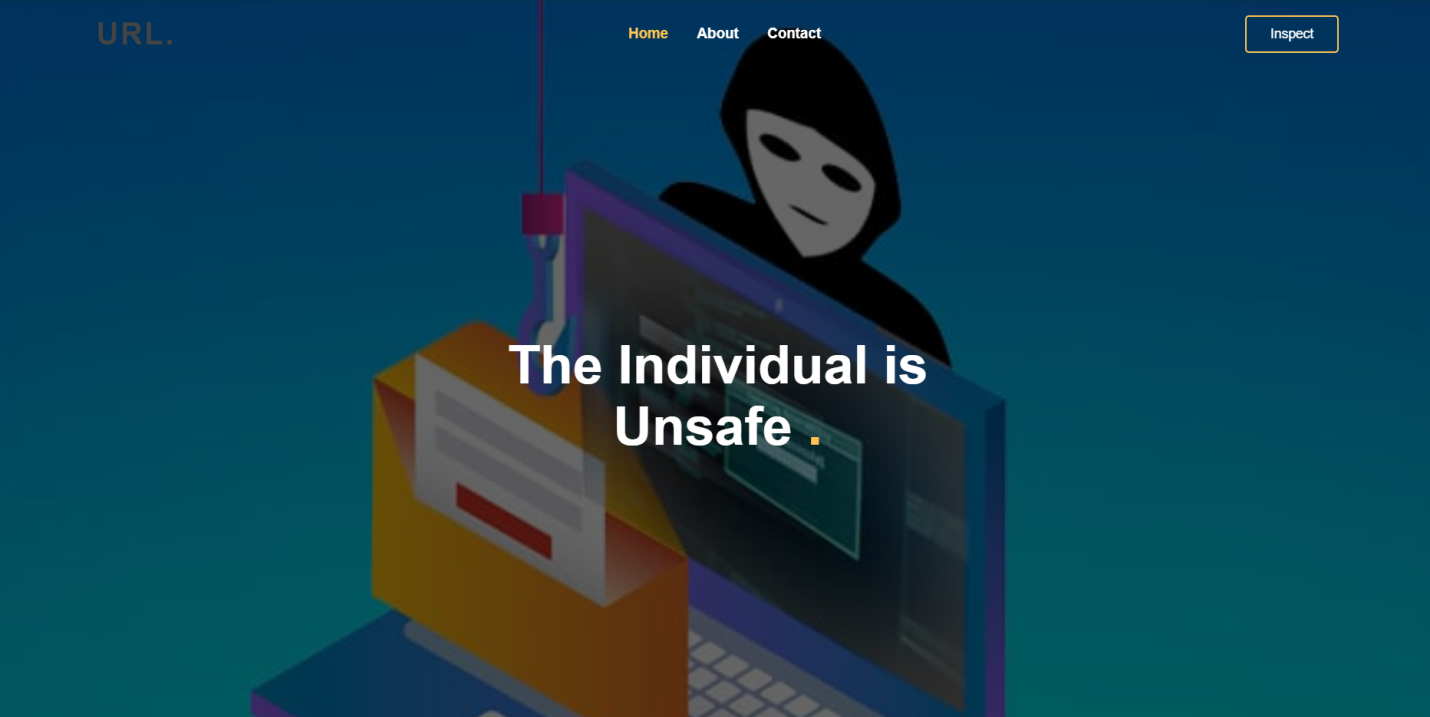








Will try with different numbers and then click on predict button.



# 7. ADVANTAGES & DISADVANTAGES

**Advantages: -**

* *Real-time Detection*: Machine learning models can analyze URLs quickly and in real-time, enabling rapid identification of phishing links, reducing the risk of falling victim to scams.
* *Scalability*: ML-based systems can handle a large number of URLs simultaneously, making them scalable to protect a vast user base or an entire organization from phishing threats.
* *Continuous Learning*: Machine learning models can adapt and improve over time by continuously learning from new phishing patterns, staying up-to-date with emerging threats.
* *Accuracy:* Advanced ML algorithms can achieve high accuracy in detecting phishing URLs, minimizing false positives and false negatives, leading to more reliable protection.
* And some other advantages are automation, customizability, early warning, multi-platform support, enhanced security, data driven insights.

**Disadvantages:**

* *False Positives and False Negatives*: Machine learning models may occasionally misclassify legitimate URLs as phishing or fail to detect sophisticated phishing attempts, leading to false positives and false negatives, respectively.
* *Data Privacy and Security*: ML-based phishing detection often involves analyzing URLs, which could raise privacy concerns if sensitive or personal data is unintentionally processed during the analysis.
* *Rapidly Evolving Phishing Techniques*: As phishing techniques evolve, ML models might struggle to keep up with the latest strategies used by attackers.
* *Dependency on Data Sources*: ML models for phishing detection depend on timely access to relevant data sources, and any disruption in data availability could impact their performance.

## 8. APPLICATIONS

URL-based phishing detection using machine learning has various practical applications, including:

1. Email Security: Integrating ML-based phishing detection in email systems helps identify and block phishing links, protecting users from clicking on malicious URLs sent via emails.

2. Web Browsers: Browser extensions or built-in features that utilize ML can alert users about potential phishing websites when they attempt to visit suspicious URLs.

3. Network Security: Employing ML models in network security systems can help detect and block phishing URLs in real-time, safeguarding users and organizations from cyber threats.

4. Mobile Security: Mobile apps can leverage ML-based phishing detection to warn users about fraudulent links, ensuring safer browsing on smartphones and tablets.

Some other applications are cloud services, anti-phishing solutions, social media platforms, anti-virus and security software.

By utilizing machine learning in URL-based phishing detection, these applications can effectively mitigate phishing risks and enhance overall cybersecurity.

## 9. CONCLUSION

1.The final take away form this project is to explore various machine learning models, perform Exploratory Data Analysis on phishing dataset and understanding their features.

2. Creating this notebook helped me to learn a lot about the features affecting the models to de tect whether URL is safe or not, also I came to know how to tuned model and how they affe ct the model performance.

3. The final conclusion on the Phishing dataset is that the some feature like "HTTTPS", "Anch orURL", "WebsiteTraffic" have more importance to classify URL is phishing URL or not.

4. Gradient Boosting Classifier correctly classify URL upto 94.52% respective classes and hen ce reduces the chance of malicious attachments.

# 10. FUTURE SCOPE

In future if we get structured dataset of phishing we can perform phishing detection much more faster than any other technique.In future we can use a combination of any other two or more classifier to get maximum accuracy.

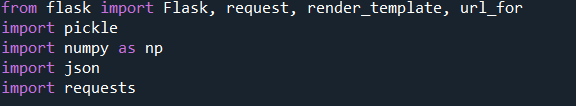
**11. BIBLIOGRAPHY**

* **ML Concepts**
  + Supervised learning: <https://www.javatpoint.com/supervised-machine-learning>
  + Unsupervised learning: <https://www.javatpoint.com/unsupervised-machine-learning>
  + Decision tree: <https://www.javatpoint.com/machine-learning-decision-tree-classification-algorithm>
  + Random forest: <https://www.javatpoint.com/machine-learning-random-forest-algorithm>
  + KNN: <https://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning>
  + Support vector machine algorithm: <https://www.javatpoint.com/machine-learning-support-vector-machine-algorithm>
  + Logistic Regression: <https://www.javatpoint.com/logistic-regression-in-machine-learning>
  + Naïve Bayes Classifier : <https://www.javatpoint.com/machine-learning-naive-bayes-classifier>
  + Gradient boosting: <https://www.javatpoint.com/gbm-in-machine-learning>
  + Multi-layer Perceptron: https://www.javatpoint.com/multi-layer-perceptron-in-tensorflow
  + Evaluation metrics: <https://www.analyticsvidhya.com/blog/2019/08/11-important-model-evaluation-error-metrics/>
* **Flask Basics** : <https://www.youtube.com/watch?v=lj4I_CvBnt0>

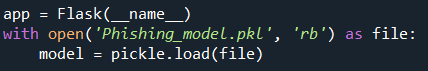
**APPENDIX**

### 11.1. Source Code

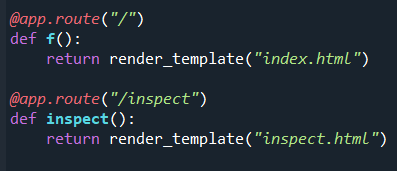
Import the libraries

****

Load the saved model. Importing flask module in the project is mandatory. An object of Flask class is our WSGI application. Flask constructor takes the name of the current module (\_\_name\_\_) as argument.

****

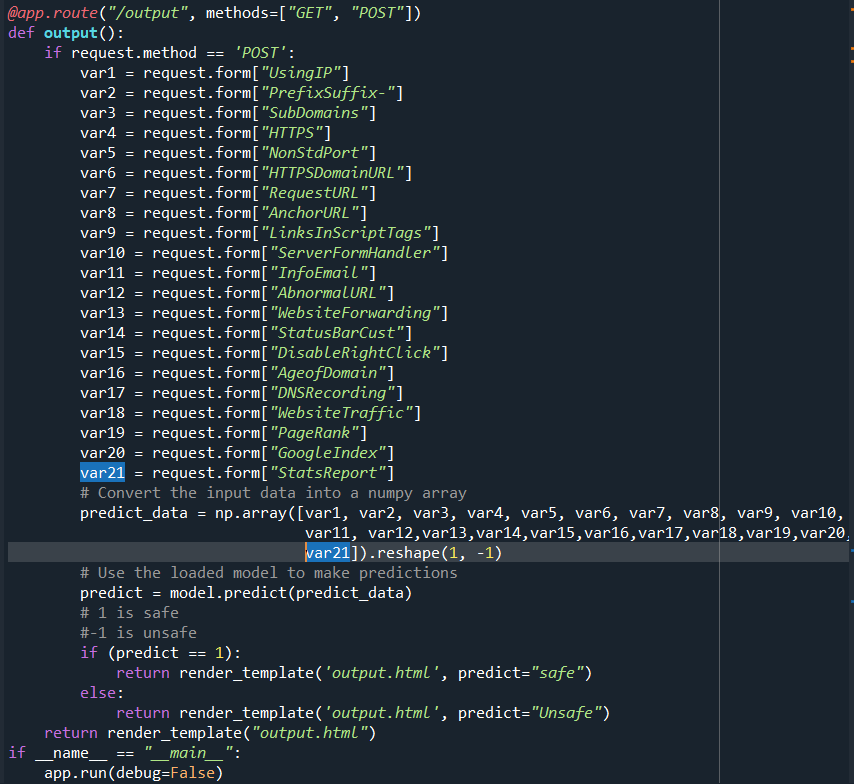
Render HTML page:



Here we will be using declared constructor to route to the HTML page which we have created earlier.

In the above example, ‘/’ URL is bound with index.html function. Hence, when the index page of the web server is opened in browser, the html page will be rendered. Whenever you enter the values from the html page the values can be retrieved using POST Method.

Retrieves the value from UI:



Here we are routing our app to output() function. This function retrieves all the values from the HTML page using Post request. That is stored in an array. This array is passed to the model.predict() function. This function returns the prediction. And this prediction value will rendered to the text that we have mentioned in the output.html page earlier.

Main function:



To run the application:

* Open anaconda prompt from the start menu
* Navigate to the folder where your python script is.
* Now type “python app.py” command
* Navigate to the localhost where you can view your web page.
* Click on the inspect button from the top right corner, enter the inputs, click on the predict button, and see the result/prediction on the web.