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

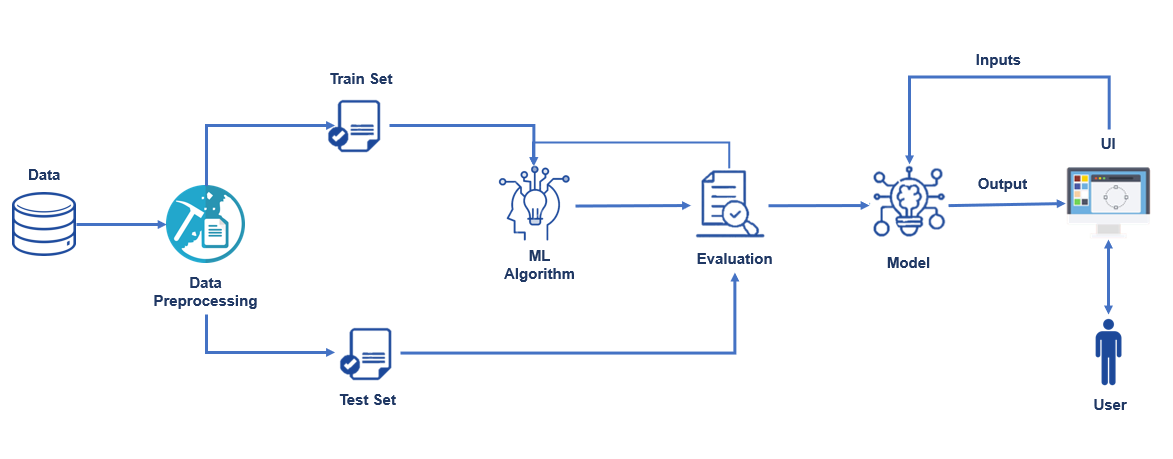
**Project Description:**

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.

This 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.

**Architecture**

# Prerequisites:

**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.

# Prior Knowledge:

You must have prior knowledge of following topics to complete this project.

* **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>

**Project Objectives:**

**By the end of this project:**

* You’ll be able to understand the problem to classify if it is a regression or a classification kind of problem.
* You will be able to know how to pre-process/clean the data using different data pre-processing techniques.
* Applying different algorithms according to the dataset
* You will be able to know how to find the accuracy of the model.
* You will be able to build web applications using the Flask framework.

**Project Flow**

* Download the dataset.
* Preprocess or clean the data.
* Analyze the pre-processed data.
* Train the machine with preprocessed data using an appropriate machine learning algorithm.
* Save the model and its dependencies.
* Build a Web application using a flask that integrates with the model built.

### Project Folder Structure

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**

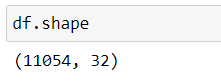
The download data set is not suitable for training the machine learning model as it might have so much of randomness so we need to clean the dataset properly in order to fetch good results. This activity includes the following steps.

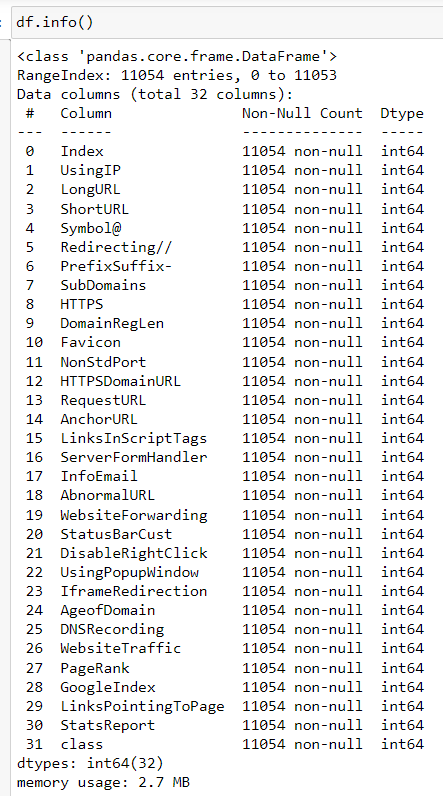
* Handling missing values
* Handling categorical data
* Handling outliers
* Scaling Techniques
* Splitting dataset into training and test set

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.

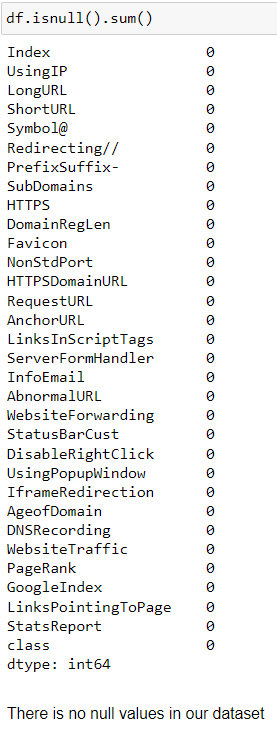
**Activity 4: Checking for null values**

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.

****

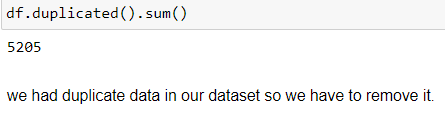
****

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.

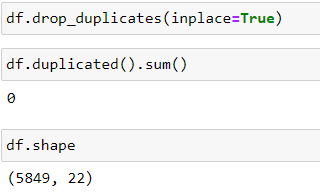


There is no categorical data in our dataset.

**Activiy 5: Checking for duplicated data**

****

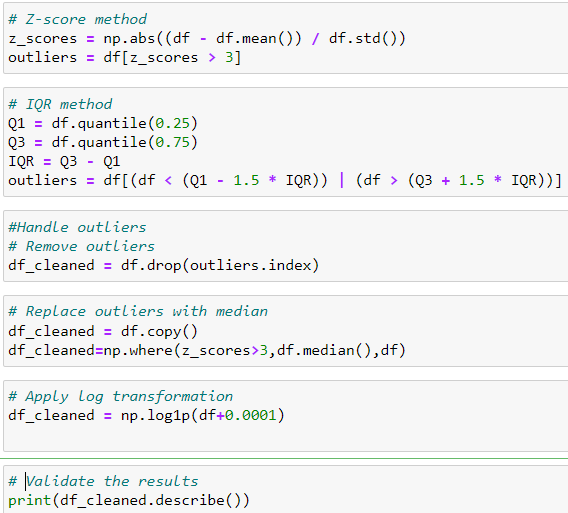
**Handling duplicate data**

****

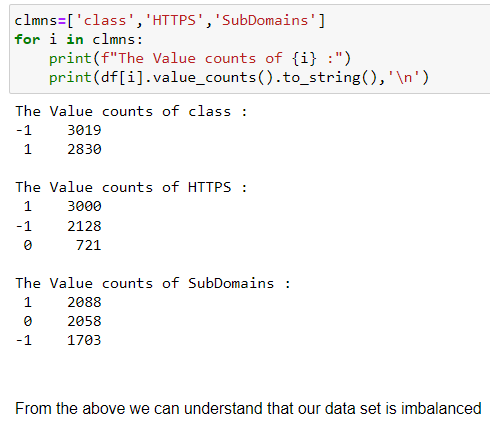
**Activity 6: Outliers**

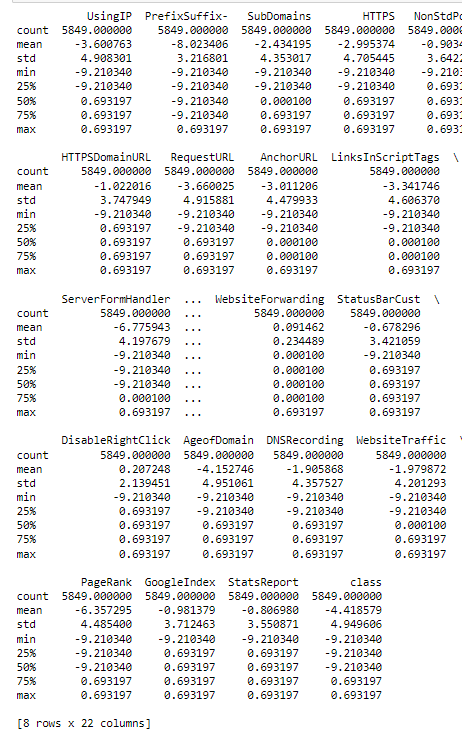
We had outliers in our dataset in the columns 'PrefixSuffix-','NonStdPort', 'HTTPSDomainURL','AnchorURL','ServerFormHandler','InfoEmail','AbnormalURL','WebsiteForwarding','StatusBarCust','DisableRightClick','GoogleIndex','StatsReport'

**Handling Outliers**

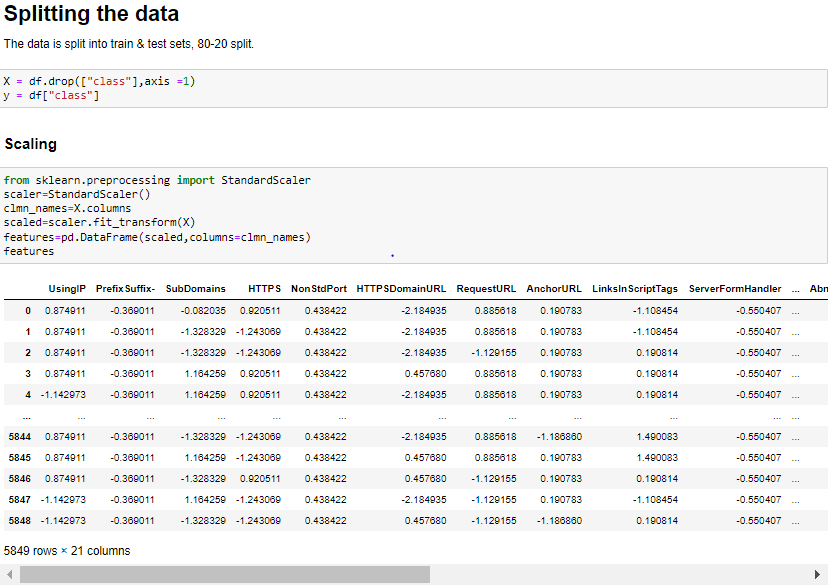
****

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

****

****

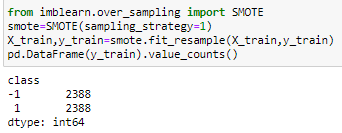
**Activity 8: Scaling**

****

### Activity 9: Train Test and Split

### 

**Activity 10: Handling Balance Data**

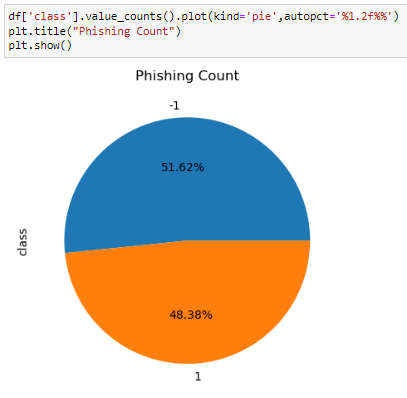
****

From the above we understand that our data is balanced.

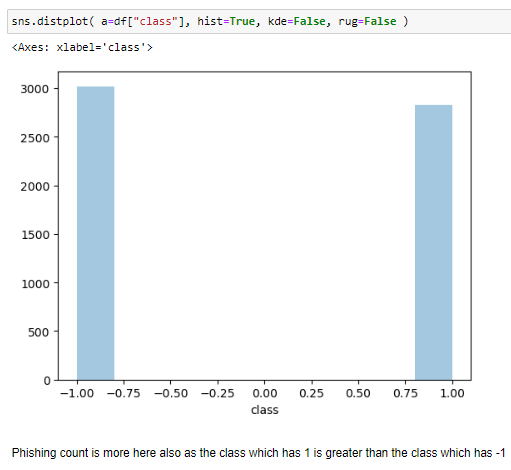
# Milestone 2: Visualizing and analysing the data

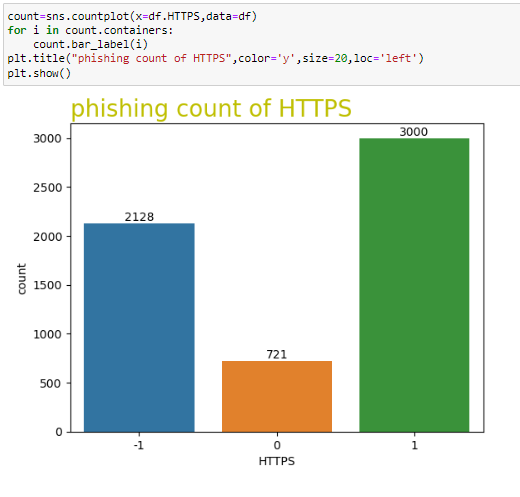
**Activity 1: Univariate analysis**

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

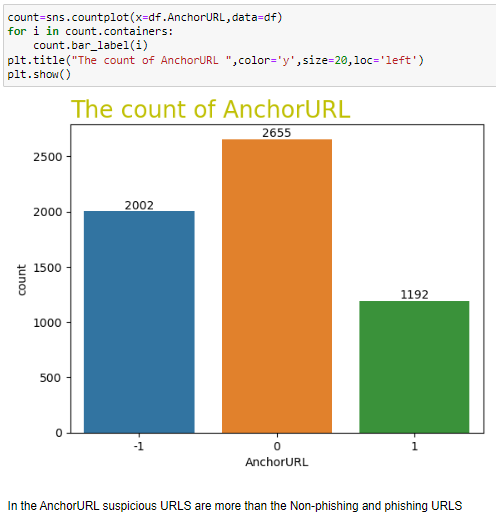
****

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



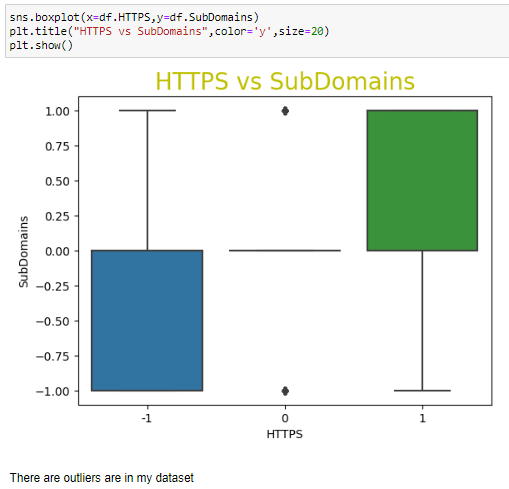


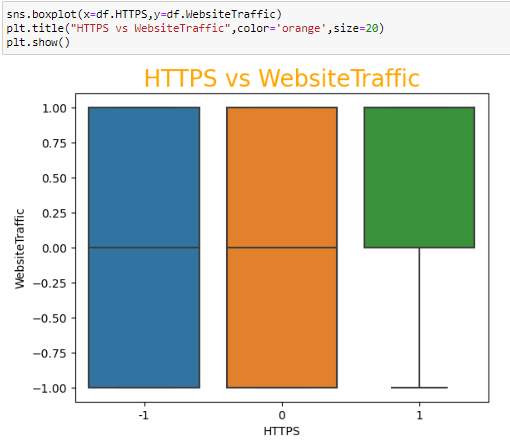
* phishing URLS are 2128.
* legitimate HTTPS URLS or non\_phishing URLS are 3000.
* Suspicious URLS are 721 (0 indicates a potential risk of features that indicate a potential risk of phishing, but they are not confirmed phishing URLS).

****

**Activity 2: Bivariate analysis**

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

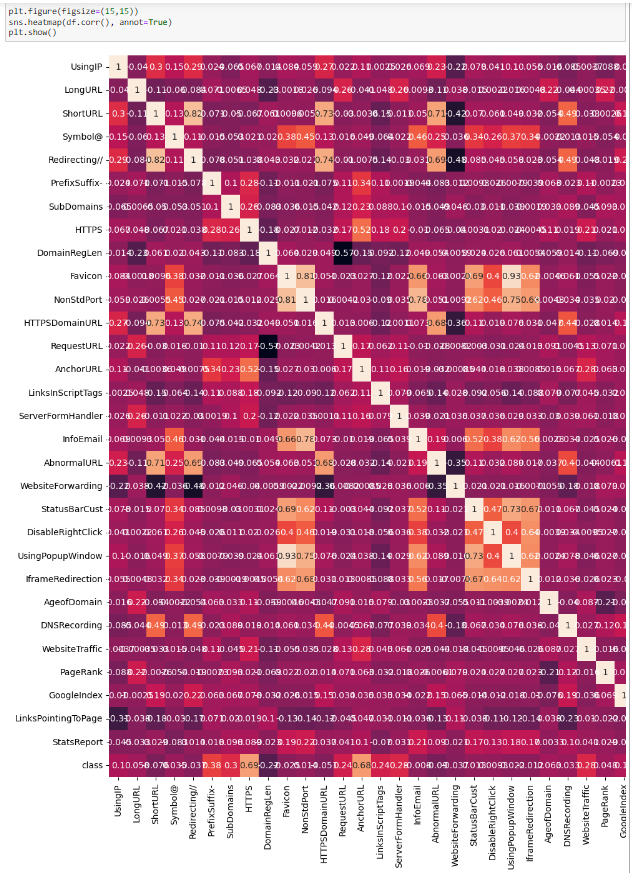
****

****

**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.

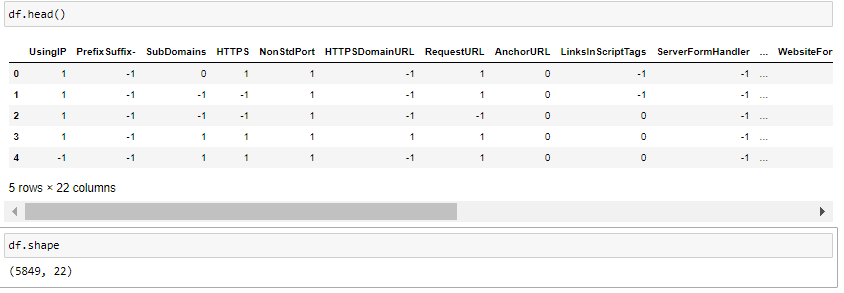
* From the below image, we came to a conclusion that how data is distributed and how they are and how much they are correlated each other.
* All the features weather following the normal distribution or not ?

****

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 columns.

# Milestone 3: Model Building

Supervised machine learning is one of the most commonly used and successful types of machine learning. Supervised learning is used whenever we want to predict a certain outcome/label from a given set of features, and we have examples of features-label pairs. We build a machine learning model from these features-label pairs, which comprise our training set. Our goal is to make accurate predictions for new, never-before-seen data.

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:

1. Logistic Regression

2. K-Nearest Neighbors

3. Naive Bayes

4. Decision Tree

5. Random Forest

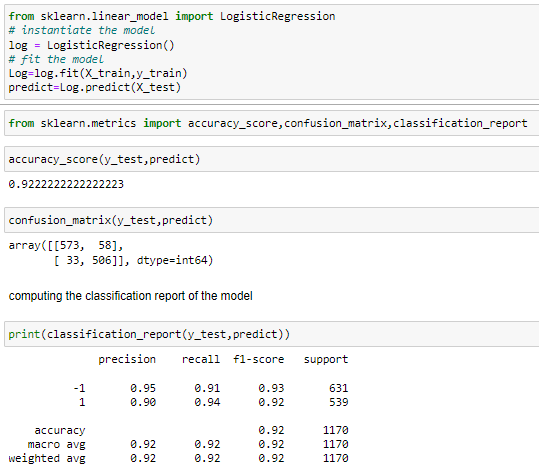
6. Gradient Boosting

7. Multi Layer perceptron Classifier

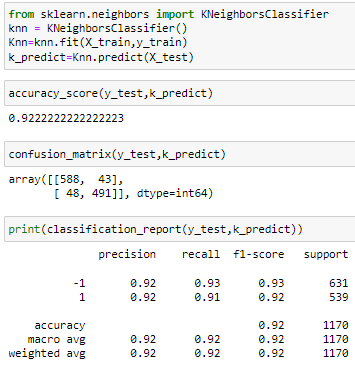
8. Support Vector Machine Classifier

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

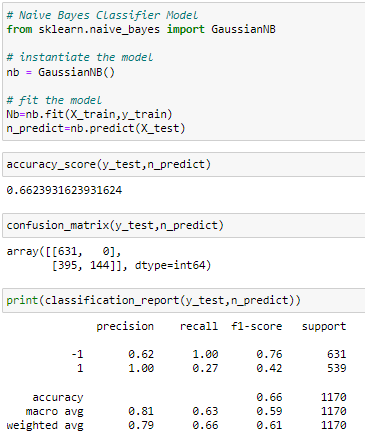
**Logistic Regression**

Logistic regression predicts the output of a categorical dependent variable. Therefore the outcome must be a categorical or discrete value. Logistic Regression is much similar to the Linear Regression except that how they are used. Linear Regression is used for solving Regression problems, whereas Logistic regression is used for solving the classification problems.

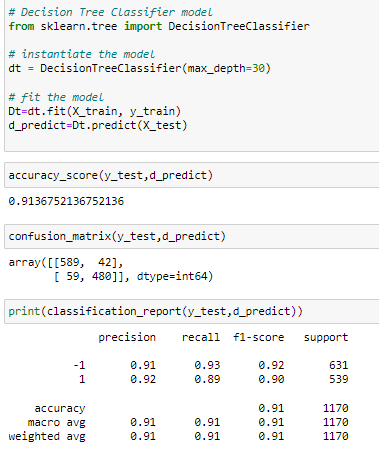
**K-Nearest Neighbors : Classifier**

K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique. K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories.

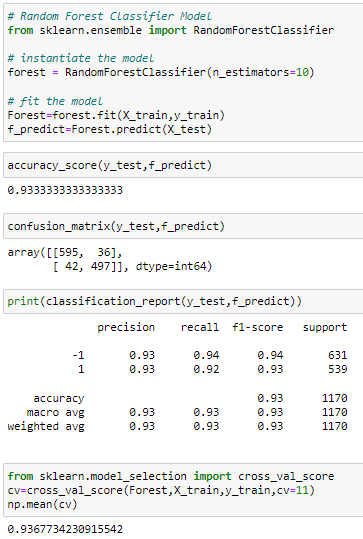
**Naive Bayes : Classifier**

Naive Bayes algorithm is a supervised learning algorithm, which is based on Bayes theorem and used for solving classification problems.It is mainly used in text, image classification that includes a high-dimensional training dataset. Naive Bayes Classifier is one of the simple and most effective Classification algorithms which helps in building the fast machine learning models that can make quick predictions.

**Decision Trees : Classifier**

Decision Tree is a Supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome.

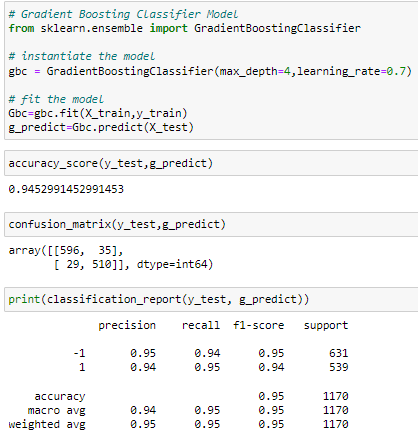
**Random Forest : Classifier**

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model.

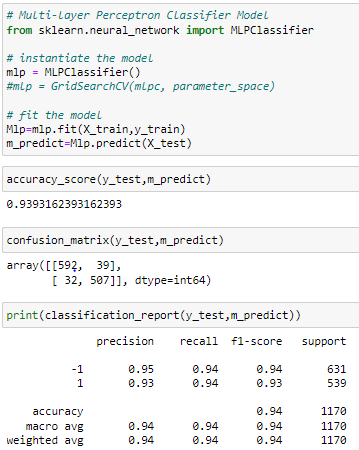
**Gradient Boosting Classifier**

Gradient boosting classifiers are a group of machine learning algorithms that combine many weak learning models together to create a strong predictive model.Decision trees are usually used when doing gradient boosting.

Boosting algorithms play a crucial role in dealing with bias variance trade-off. Unlike bagging algorithms, which only controls for high variance in a model, boosting controls both the aspects (bias & variance), and is considered to be more effective.

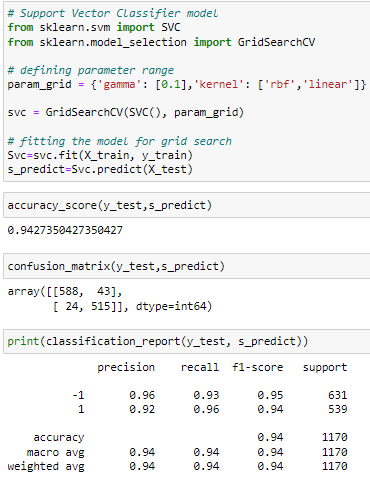


**Multi-layer Perceptron classifier**

MLPClassifier stands for Multi-layer Perceptron classifier which in the name itself connects to a Neural Network. Unlike other classification algorithms such as Support Vectors or Naive Bayes Classifier, MLPClassifier relies on an underlying Neural Network to perform the task of classification.

**Support Vector Machine : Classifier**

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems.

The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future.

**Mile stone -4: Comparision of Models**

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

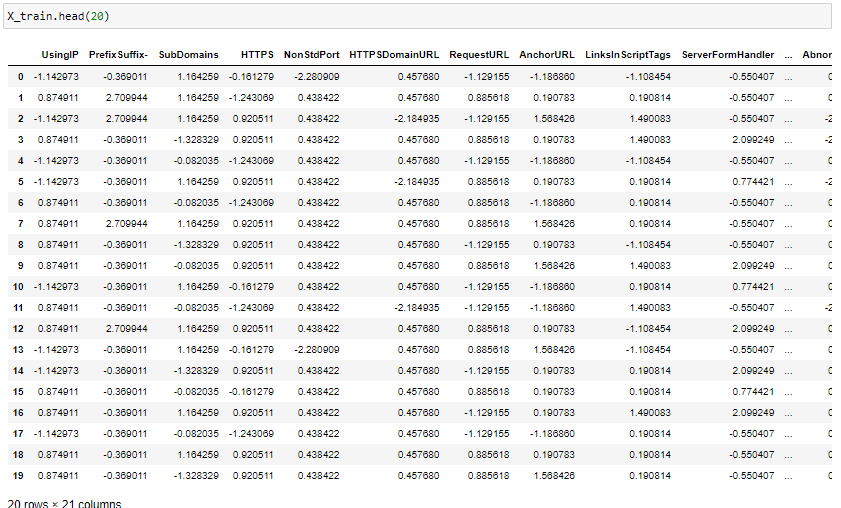
accuracy 0.66 1170

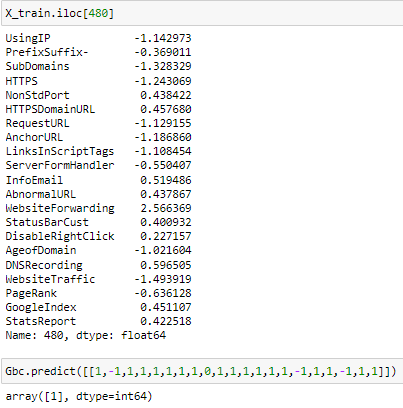
macro avg 0.81 0.63 0.59 1170

weighted avg 0.79 0.66 0.61 1170

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

**Testing the model:**

****

****

**Milestone 5: Storing Best Model**

****

**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 detect whether URL is safe or not, also I came to know how to tuned model and how they affect the model performance.

3. The final conclusion on the Phishing dataset is that the some feature like "HTTTPS", "AnchorURL", "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 hence reduces the chance of malicious attachments.

# Milestone 6: Application Building

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

This section has the following tasks

* Building HTML Pages
* Building serverside script

**Activity1: Building Html Pages:**

For this project create three HTML files namely

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

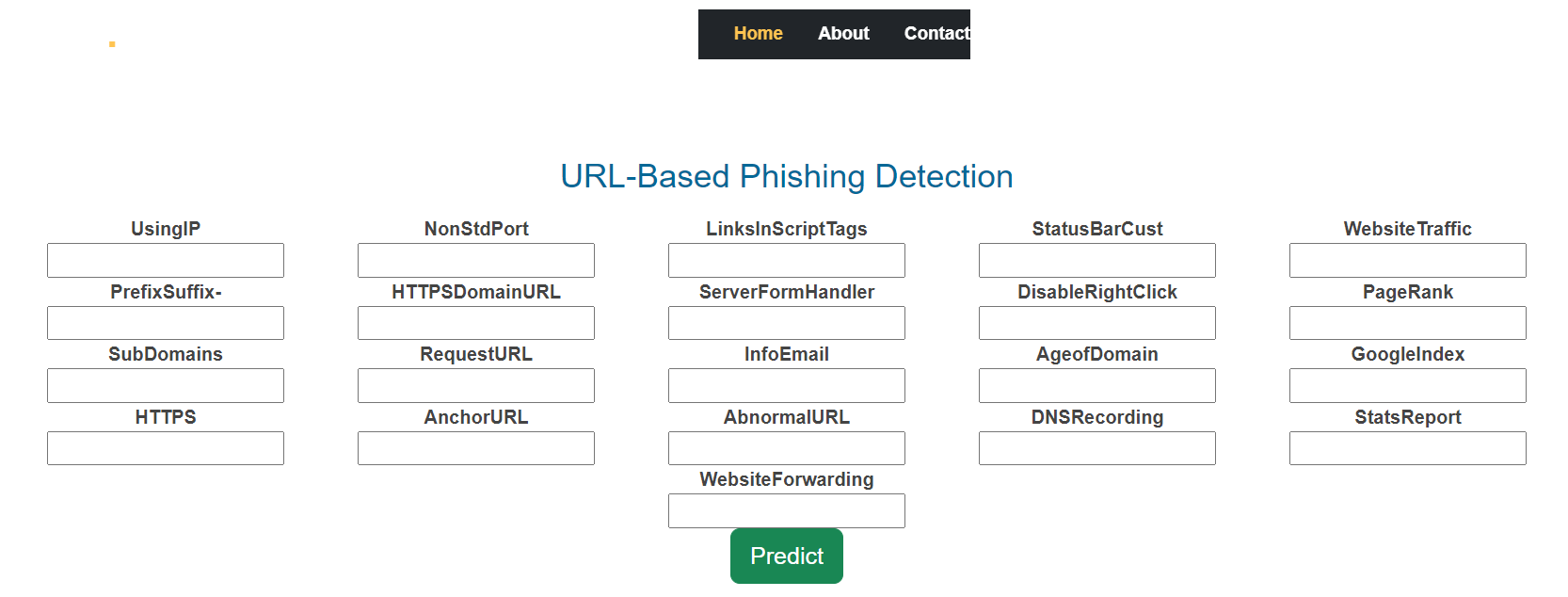
and save them in templates folder.

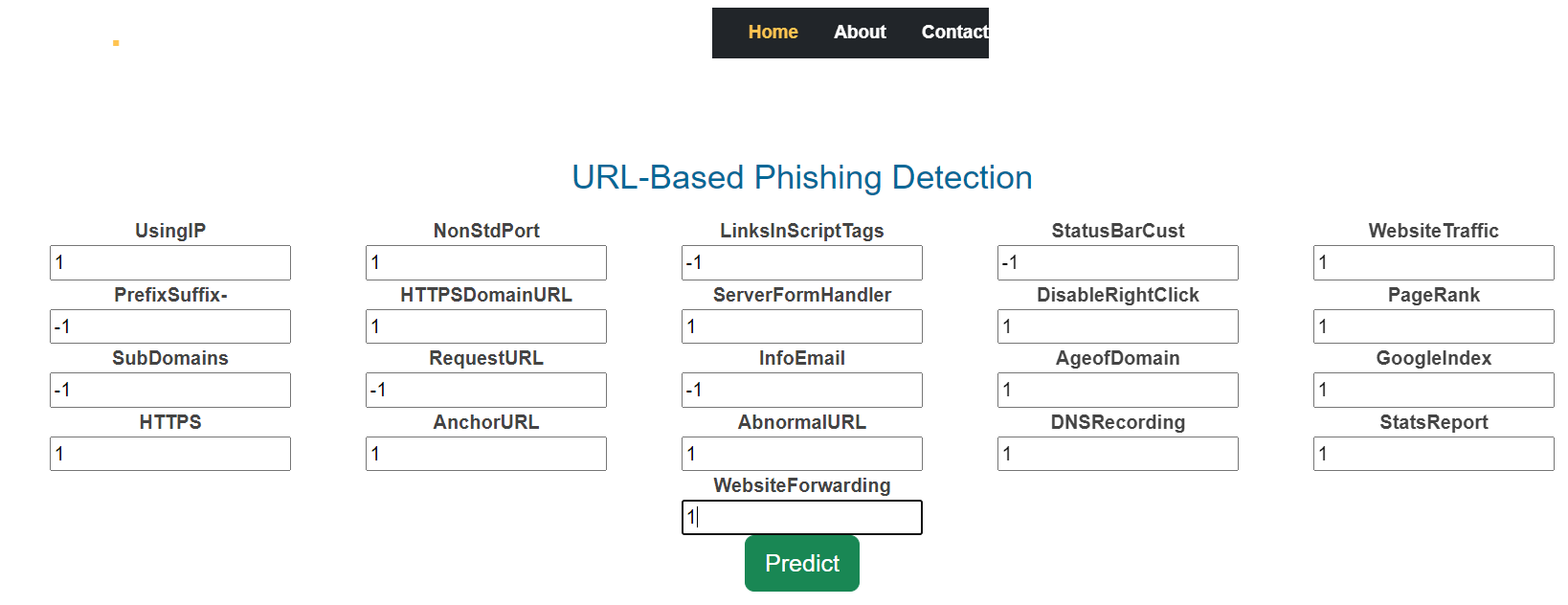
This is how our index.html page looks like:



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

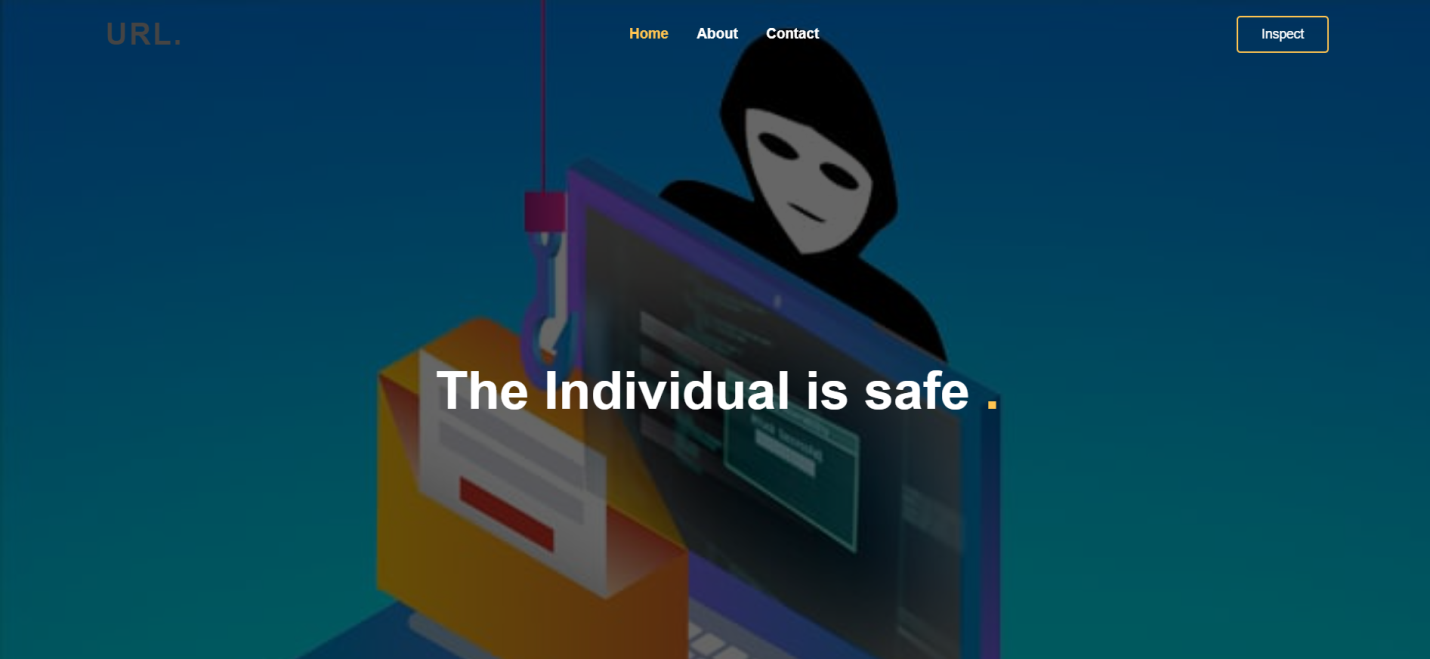
Lets look how our Inspect.html file looks like:

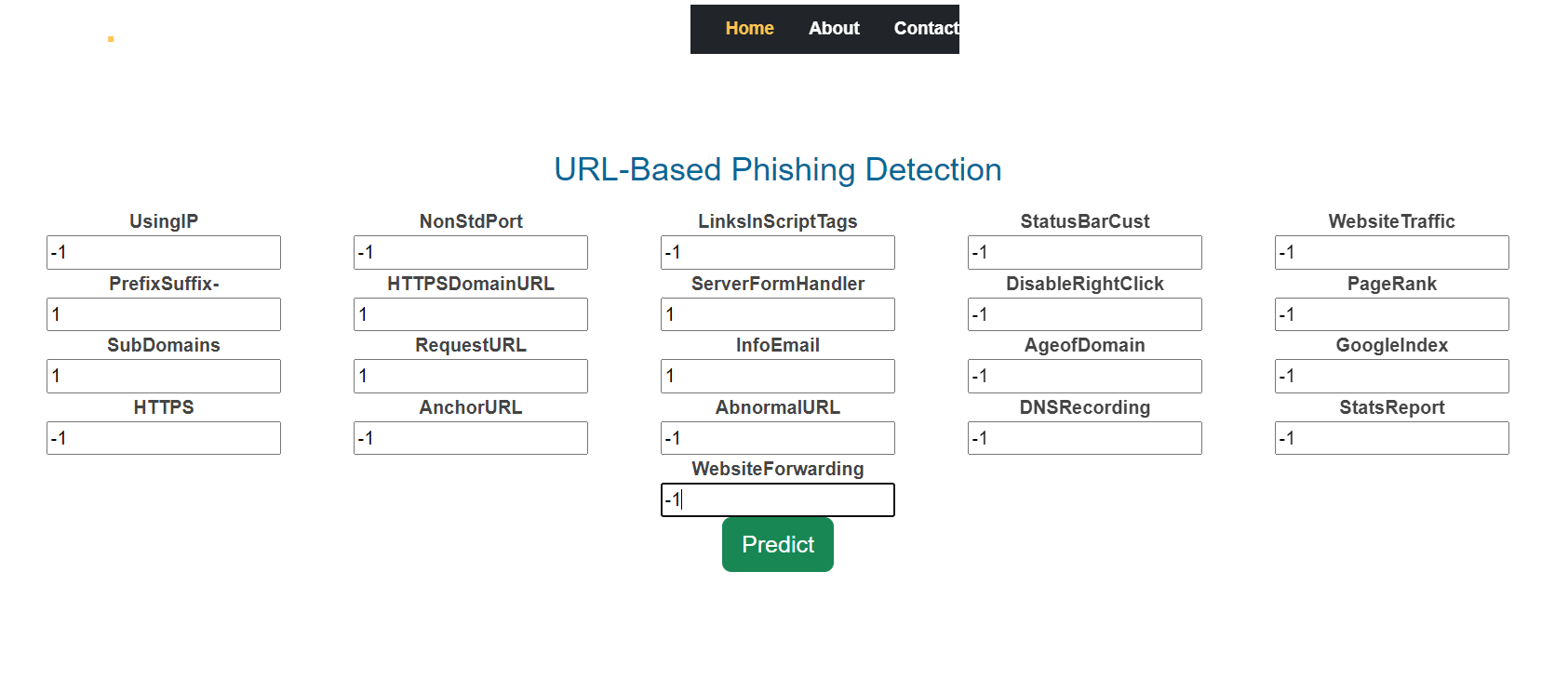




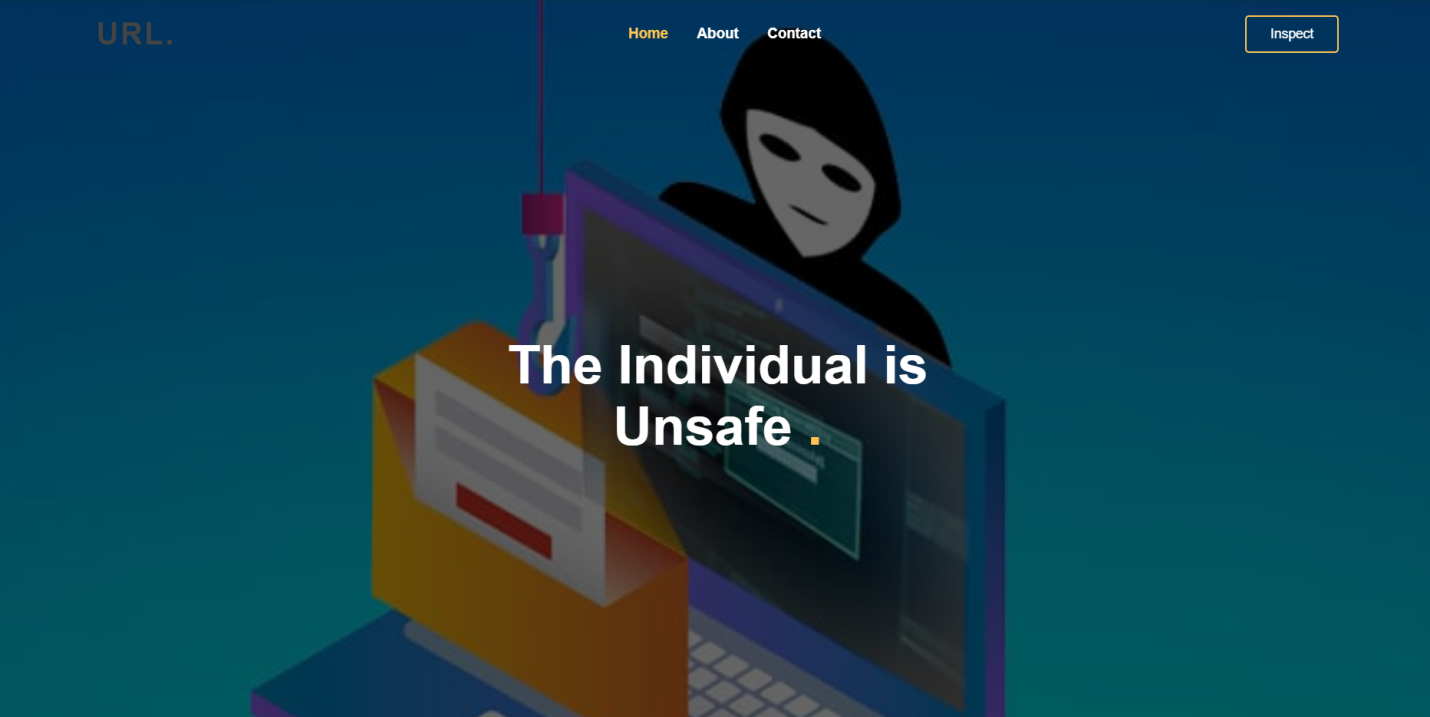
Now when you click on predict button you will get redirected to output.html

Lets look how our output.html file looks like:



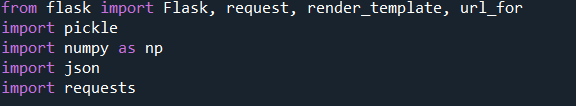


Will try with different numbers and then click on predict button.

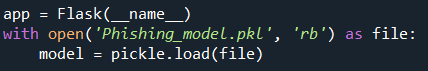


**Activity2: Build Python code:**

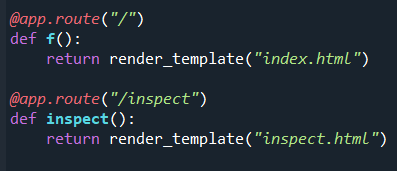
Import the libraries

****

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.

****

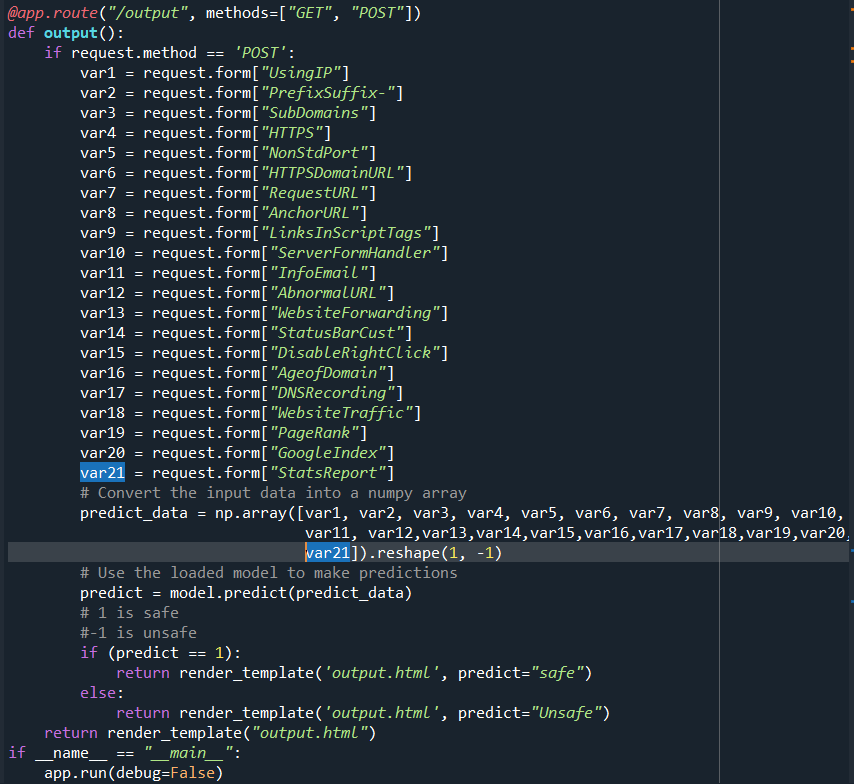
Render HTML page:



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:



**Activity3: 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.