Optimizing Spam Filtering With Machine Learning

**Project Description:**

Over recent years, as the popularity of mobile phone devices has increased, Short Message Service (SMS) has grown into a multi-billion dollar industry. At the same time, reduction in the cost of messaging services has resulted in growth in unsolicited commercial advertisements (spams) being sent to mobile phones. Due to Spam SMS, Mobile service providers suffer from some sort of financial problems as well as it reduces calling time for users. Unfortunately, if the user accesses such Spam SMS they may face the problem of virus or malware. When SMS arrives at mobile it will disturb mobile user privacy and concentration. It may lead to frustration for the user. So Spam SMS is one of the major issues in the wireless communication world and it grows day by day.

To avoid such Spam SMS people use white and black list of numbers. But this technique is not adequate to completely avoid Spam SMS. To tackle this problem it is needful to use a smarter technique which correctly identifies Spam SMS. Natural language processing technique is useful for Spam SMS identification. It analyses text content and finds patterns which are used to identify Spam and Non-Spam SMS.

**Technical Architecture:**



**Project Flow:**

User interacts with the UI to enter the input.

Entered input is analysed by the model which is integrated.

Once model analyses the input the prediction is showcased on the UI

To accomplish this, we have to complete all the activities listed below,

Define Problem / Problem Understanding

Specify the business problem

Business requirements

Literature Survey

Social or Business Impact.

Data Collection & Preparation

Collect the dataset

Data Preparation

Exploratory Data Analysis

Descriptive statistical

Visual Analysis

Model Building

Training the model in multiple algorithms

Testing the model

Performance Testing & Hyperparameter Tuning

Testing model with multiple evaluation metrics

Comparing model accuracy before & after applying hyperparameter tuning

Model Deployment

Save the best model

Integrate with Web Framework

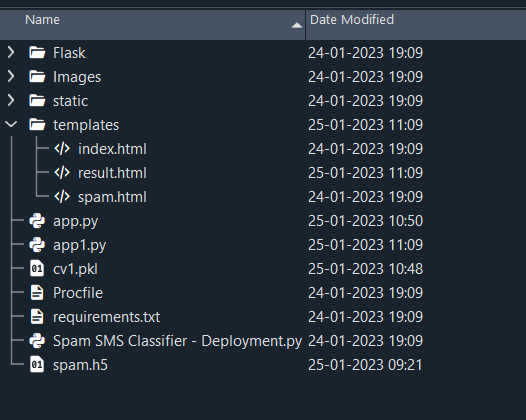
Project Demonstration & Documentation

Record explanation Video for project end to end solution

Project Documentation-Step by step project development procedure

Project 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 and a python script app.py for scripting.

Spam.h5 is our saved model. Further we will use this model for flask integration.

**Define Problem / Problem Understanding**

In this milestone, we will see the define problem and problem understanding.

**Specify The Business Problem**

Over recent years, as the popularity of mobile phone devices has increased, Short Message Service (SMS) has grown into a multi-billion dollar industry. At the same time, reduction in the cost of messaging services has resulted in growth in unsolicited commercial advertisements (spams) being sent to mobile phones. Due to Spam SMS, Mobile service providers suffer from some sort of financial problems as well as it reduces calling time for users. Unfortunately, if the user accesses such Spam SMS they may face the problem of virus or malware. When SMS arrives at mobile it will disturb mobile user privacy and concentration. It may lead to frustration for the user. So Spam SMS is one of the major issues in the wireless communication world and it grows day by day.

**Business Requirements**

A business requirement for an SMS spam classification system would include the ability to accurately identify and flag spam messages, protect customers from unwanted or harmful messages, and comply with industry regulations and laws regarding spam messaging. Additionally, the system should be able to handle a high volume of messages, integrate with existing systems and databases, and provide reporting and analysis capabilities to track performance and improve the system over time. The system should also have an easy-to-use interface and be easy to maintain and update.

**Literature Survey**

project would involve researching and analysing existing studies, papers, and articles on the topic to gain a thorough understanding of the current state of SMS spam classification and to identify potential areas for improvement and future research. The survey would include looking at different methods and techniques used for identifying and flagging spam messages, such as machine learning algorithms, natural language processing, and rule-based systems. It would also involve evaluating the performance and effectiveness of

these methods, as well as their limitations and challenges. Additionally, the literature survey would review the current state of SMS spam and trends in the industry, as well as any existing laws and regulations related to spam messaging. The survey would also investigate the datasets and feature representations used in previous studies, which would help to determine the best approach for the current project. Furthermore, It would be important to check the pre-processing techniques used in the research to understand how to properly clean and prepare the data for the classifier

**Social Or Business Impact**

Social Impact:- it can help protect individuals from unwanted and potentially harmful messages. Spam messages can include phishing attempts, scams, and fraud, which can have serious financial and personal consequences for recipients. By accurately identifying and flagging spam messages, the system can help prevent these types of attacks and protect individuals from falling victim to them.

Business Model/Impact:- it can help protect their customers and improve their reputation. Spam messages can harm a business's reputation and lead to customer complaints and lost business. By accurately identifying and flagging spam messages, the system can help protect businesses and improve their customer's trust.

**Data Collection & Preparation**

ML depends heavily on data. It is the most crucial aspect that makes algorithm training possible. So this section allows you to download the required dataset.

**Collect The Dataset**

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

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

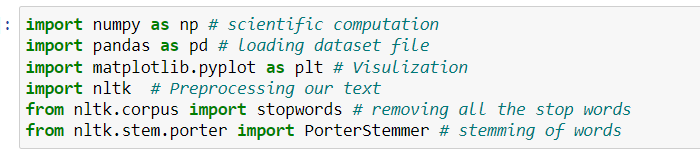
Link: https://www.kaggle.com/datasets/uciml/sms-spam-collection-dataset

As the dataset is downloaded. Let us read and understand the data properly with the help of some visualisation techniques and some analysing techniques.

Note: There are a number of techniques for understanding the data. But here we have used some of it. In an additional way, you can use multiple techniques.

**Importing The Libraries**

Import the necessary libraries as shown in the image. (optional) Here we have used visualisation style as fivethirtyeight.



**Read The 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

give the directory of the csv file.

Performance Testing & Hyperparameter Tuning

Model Deployment

Project Demonstration & Documentation

**Data Preparation**

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 in order to fetch good results. This activity includes the following steps.

Handling missing values

Handling categorical data

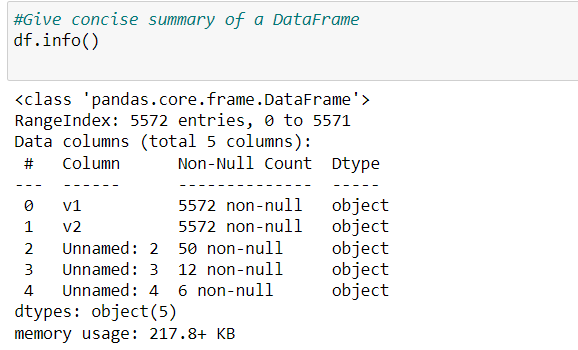
Handling Imbalance Data

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.

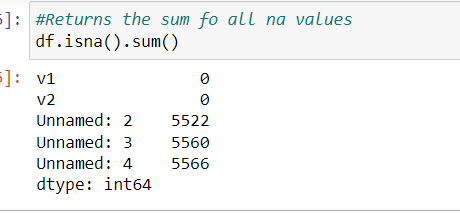
Documentation

**Handling Missing Values**

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

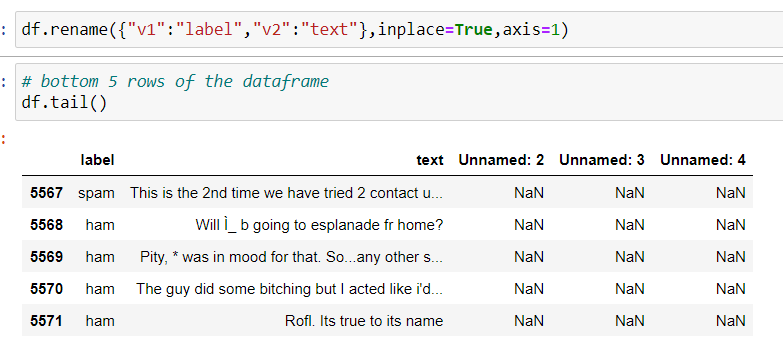


For checking the null values, df.isnull() function is used. To sum those null values we use .sum() function. From the below image we found that there are no null values present in our dataset. So we can skip handling the missing values step.



From the above code of analysis, we can infer that columns such as V1 and v2 are not having missing columns,unnamed columns are not required for analysis

Renaming the columns according the requirement



**Handling Categorical Values**

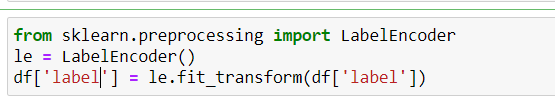
As we can see our dataset has categorical data we must convert the categorical data to integer encoding or binary encoding.

To convert the categorical features into numerical features we use encoding techniques.

There are several techniques but in our project we are using manual encoding with the help of list comprehension.

In our project,we have text column so we will be using natural language processing for processing the data. Output column is having classes we

Converting into 0 and 1 by applying label encoding

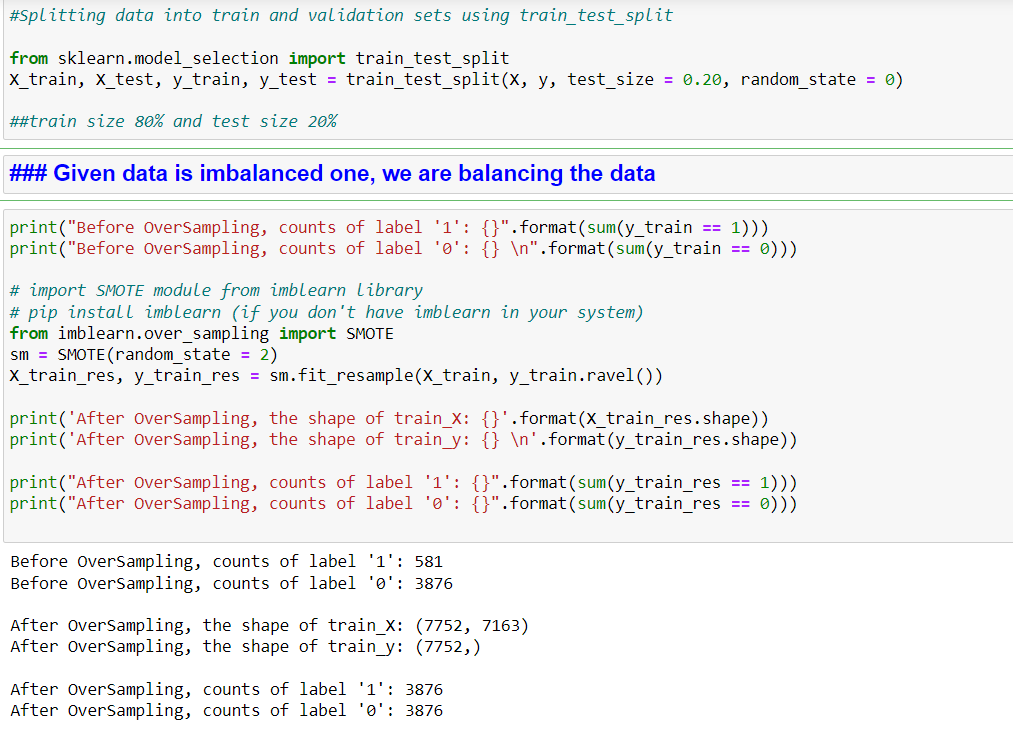


**Handling Imbalance Data**

Data Balancing is one of the most important step, which need to be performed for classification models, because when we train our model on imbalanced dataset ,we will get biassed results, which means our model is able to predict only one class element .

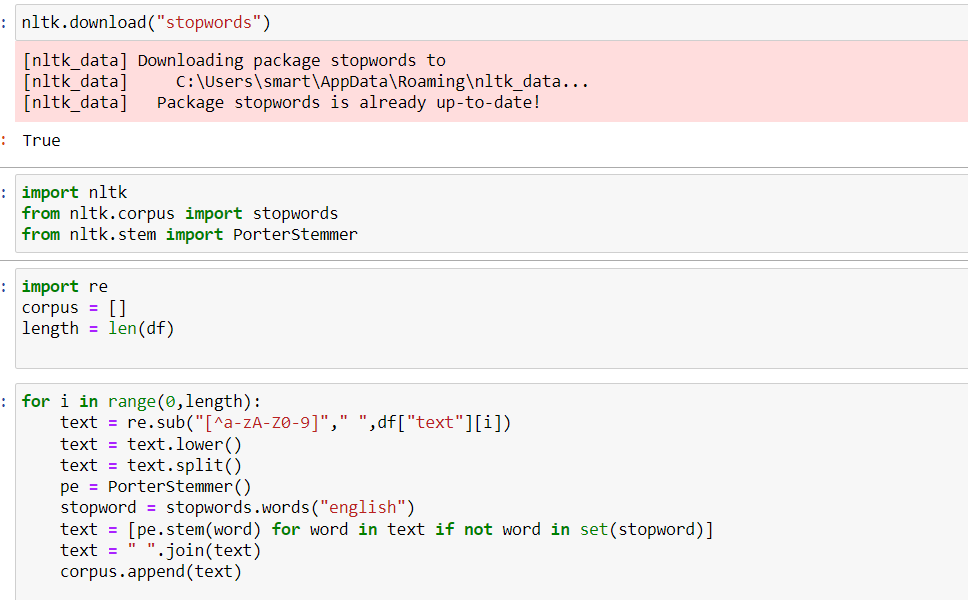
For Balancing the data we are using the SMOTE Method.

SMOTE: Synthetic minority over sampling technique, which will create new synthetic data points for under class as per the requirements given by us using KNN method.



From the above picture, we can infer that ,previously our dataset had 581 class 1, and 3876 class 0 items, after applying smote technique on the dataset the size has been changed for minority class.

**Cleaning The Text Data**



Text pre-processing includes

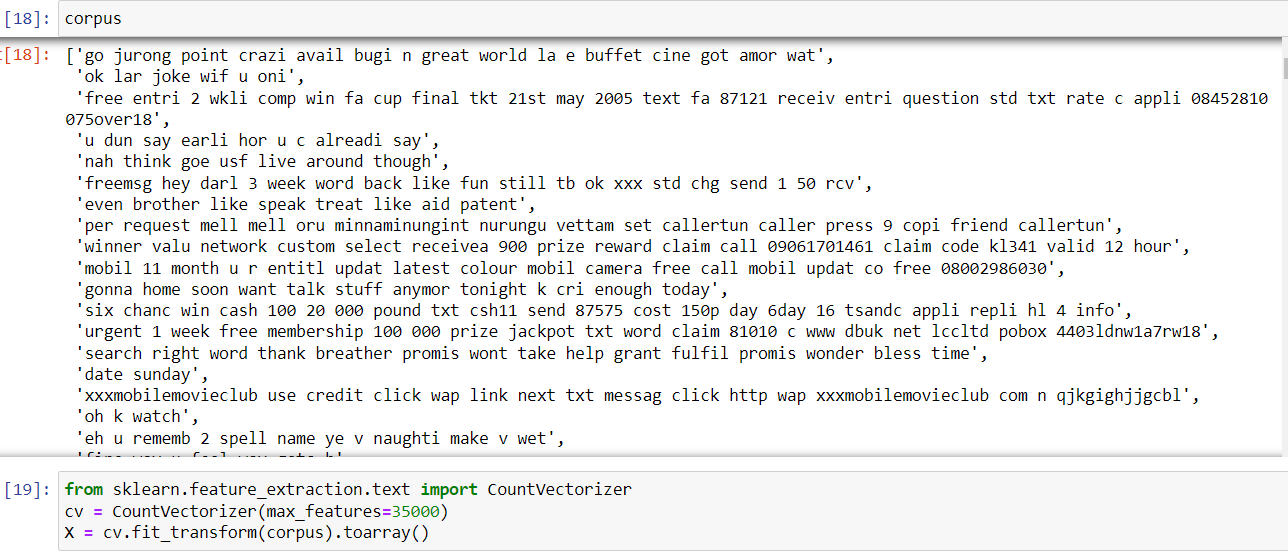
Removing punctuation from the text using regular expression library

Converting the sentence into lower case

Tokenization – splitting the sentence into words

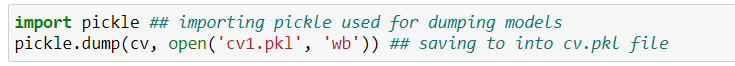
Removing stop words from the data

Stemming – stemming is the process of brining all the words into base form



After applying all the above functions, we will get corpus

Converting the corpus into Document Term matrix using Count vectorizer



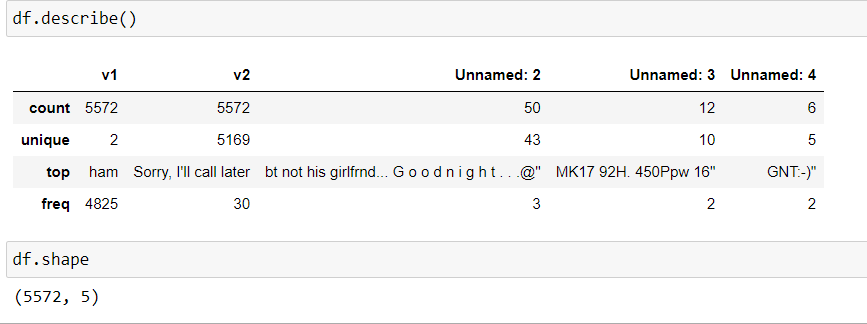
Saving the count vectorizer function for future use

**Exploratory Data Analysis**

In this milestone, we will see the exploratory data anlysis.

**Descriptive Statistical**

Descriptive analysis is to study the basic features of data with the statistical process. Here pandas has 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.



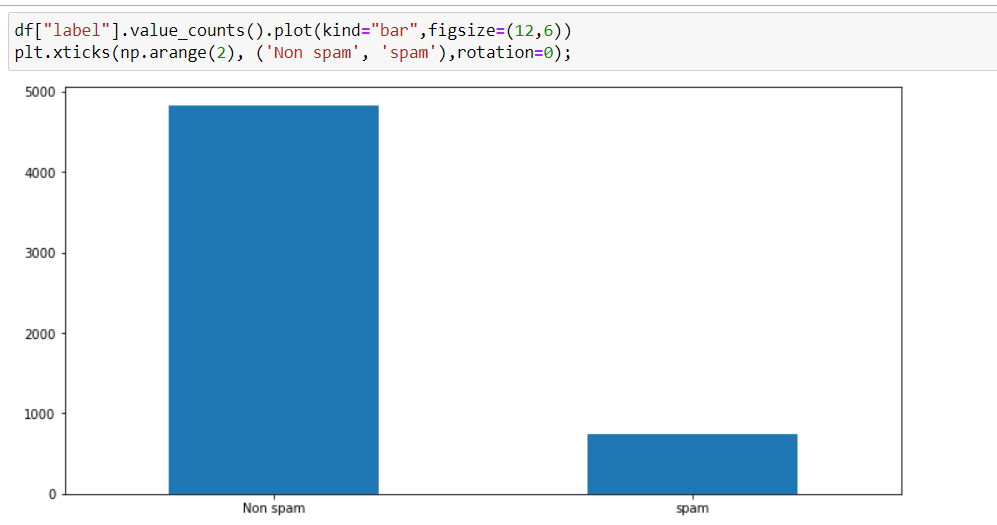
**Visual Analysis**

Visual analysis is the process of using visual representations, such as charts, plots, and graphs, to explore and understand data. It is a way to quickly identify patterns, trends, and outliers in the data, which can help to gain insights and make informed decisions.

**Univariate Analysis**

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

The Seaborn package provides a wonderful function distplot. With the help of distplot, we can find the distribution of the feature. To make multiple graphs in a single plot, we use subplot.



In our dataset we have some categorical features. With the count plot function, we are going to count the unique category in those features.

Countplot:-

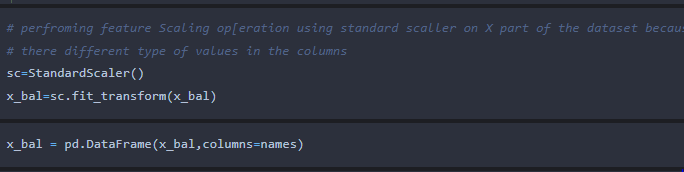
A count plot can be thought of as a histogram across a categorical, instead of quantitative, variable. The basic API and options are identical to those for barplot() , so you can compare counts across nested variables.

From the graph we can infer that , more data belongs class 0 than class 1

Scaling the Data

Scaling is one the important process, we have to perform on the dataset, because of data measures in different ranges can leads to mislead in prediction

Models such as KNN, Logistic regression need scaled data, as they follow distance based method and Gradient Descent concept.



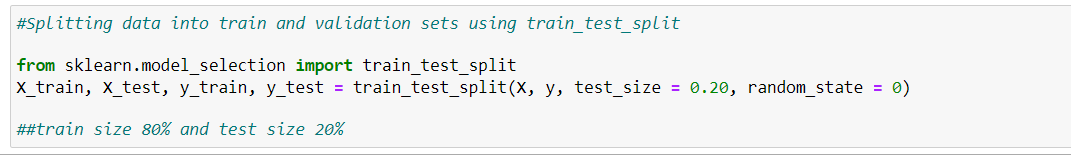
We will perform scaling only on the input values.Once the dataset is scaled, it will be converted into an array and we need to convert it back to a dataframe.

Splitting data into train and test

Now let’s split the Dataset into train and test sets

Changes: first split the dataset into x and y and then split the data set

Here x and y variables are created. On x variable, df is passed with 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.



**Model Building**

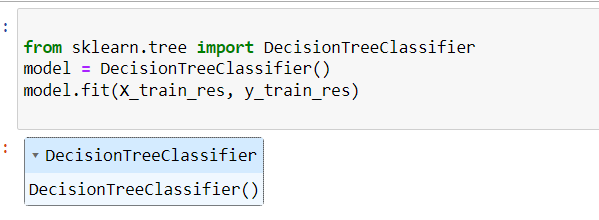
In this milestone, we will see the model building.

**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 we are applying four classification algorithms. The best model is saved based on its performance.

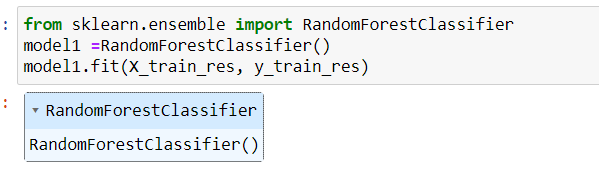
**Decision Tree Model**

A function named decisionTree is created and train and test data are passed as the parameters. Inside the function, DecisionTreeClassifier algorithm is initialised and training data is passed to the model with the .fit() function. Test data is predicted with .predict() function and saved in a new variable. For evaluating the model, a confusion matrix and classification report is done.



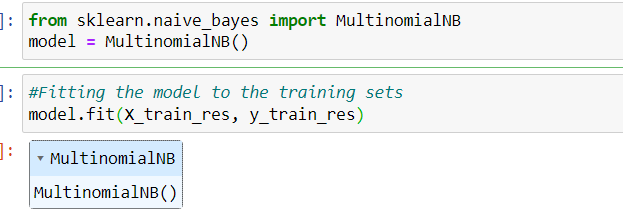
**Random Forest Model**

A function named randomForest is created and train and test data are passed as the parameters. Inside the function, RandomForestClassifier algorithm is initialised and training data is passed to the model with .fit() function. Test data is predicted with .predict() function and saved in a new variable. For evaluating the model, a confusion matrix and classification report is done.



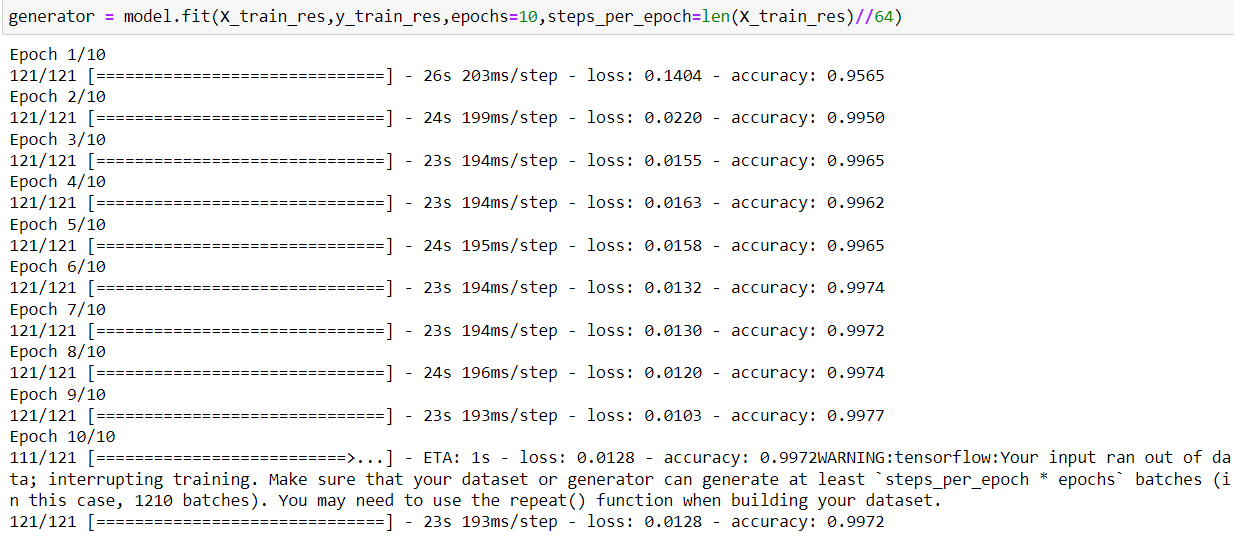
Naive Bayes Model

A function named MultinomialNB is created and train and test data are passed as the parameters. Inside the function, MultinomialNB algorithm is initialised and training data is passed to the model with .fit() function. Test data is predicted with .predict() function and saved in new variable. For evaluating the model, confusion matrix and classification report is done.

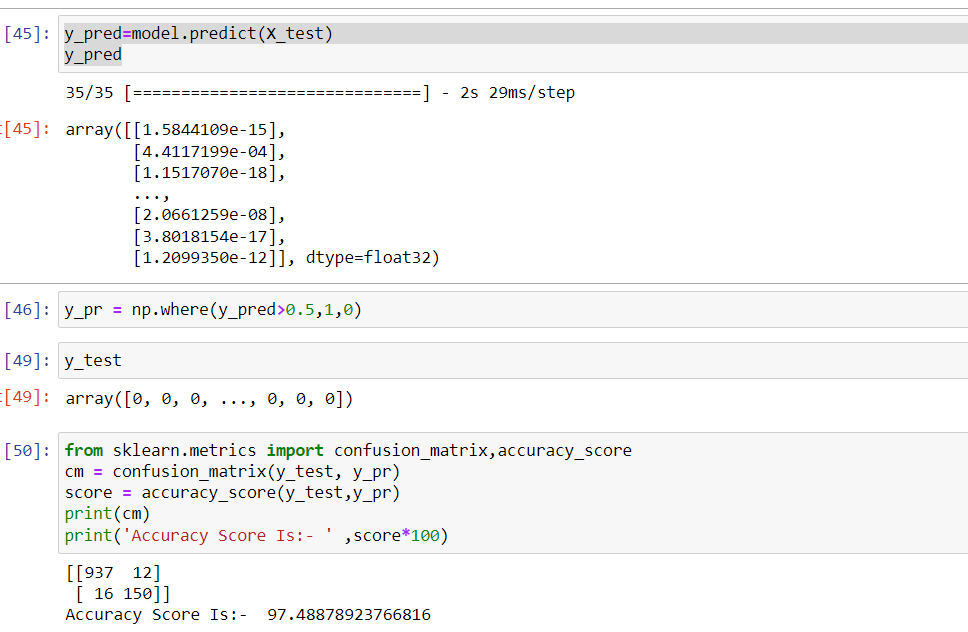


**ANN Model**

Building and training an Artificial Neural Network (ANN) using the Keras library with TensorFlow as the backend. The ANN is initialised as an instance of the Sequential class, which is a linear stack of layers. Then, the input layer and two hidden layers are added to the model using the Dense class, where the number of units and activation function are specified. The output layer is also added using the Dense class with a sigmoid activation function. The model is then compiled with the Adam optimizer, binary cross-entropy loss function, and accuracy metric. Finally, the model is fit to the training data with a batch size of 100, 20% validation split, and 100 epochs.

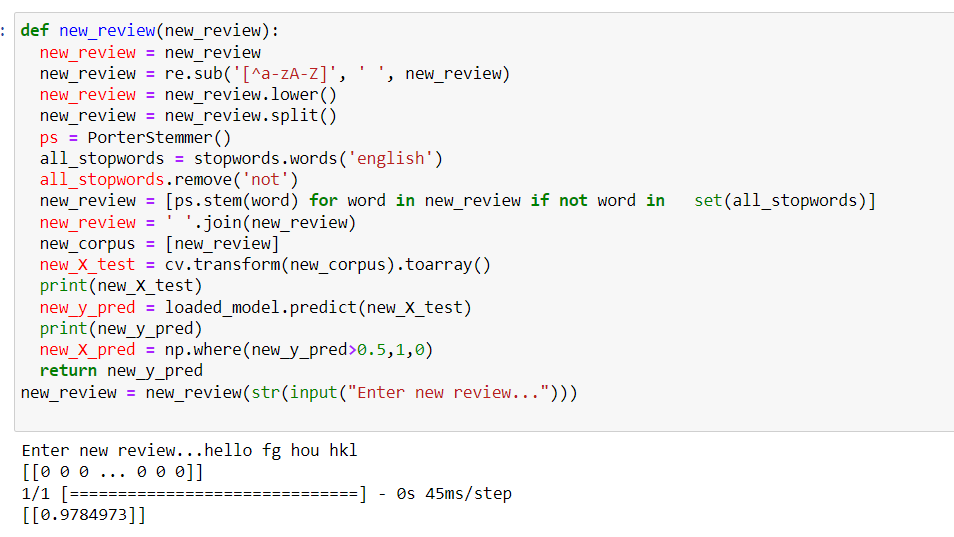


**Testing The Model**



In ANN we first have to save the model to the test the inputs

This code defines a function named "new\_review" which takes in a new\_review as an input. The function then converts the input new\_review from a list to a numpy array. It reshapes the new\_review array as it contains only one record. Then, it applies feature scaling to the reshaped new\_review array using a scaler object 'sc' that should have been previously defined and fitted. Finally, the function returns the prediction of the classifier on the scaled new\_review



**Performance Testing & Hyperparameter Tuning**

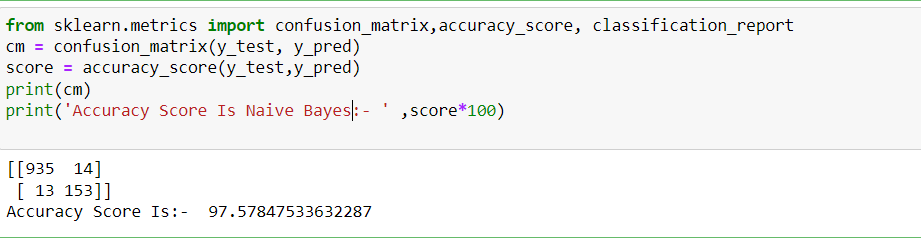
In this milestone, we will see the performance testing and hyperparameter turning.

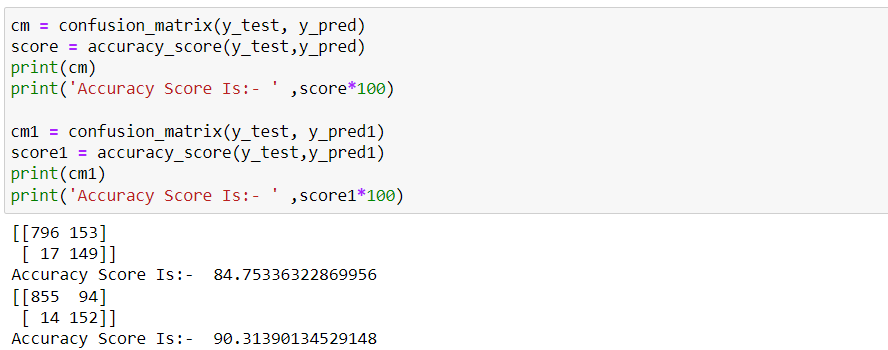
**Testing Model With Multiple Evaluation Metrics**

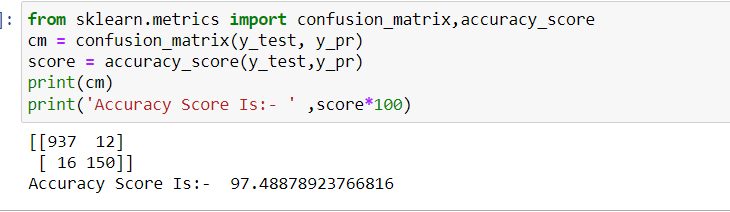
Multiple evaluation metrics means evaluating the model's performance on a test set using different performance measures. This can provide a more comprehensive understanding of the model's strengths and weaknesses. We are using evaluation metrics for classification tasks including accuracy, precision, recall, support and F1-score.

**Compare The Model**

For comparing the above four models, the compareModel function is defined.





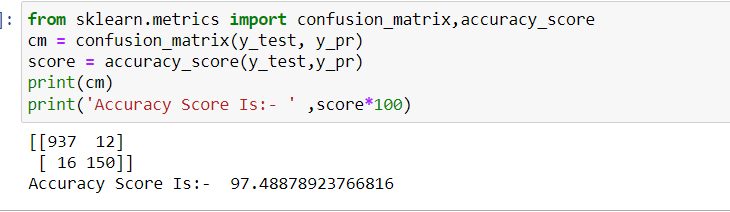


After calling the function, the results of models are displayed as output. From the five models ANN is performing well. From the below image, We can see the accuracy of the model. ANN is giving the 99.72% accuracy for the training data and 97.48 for testing data.

**Comparing Model Accuracy Before & After Applying Hyperparameter Tuning**

Evaluating performance of the model From sklearn, cross\_val\_score is used to evaluate the score of the model. On the parameters, we have given rf (model name), x, y, cv (as 5 folds). Our model is performing well. So, we are saving the model by model.save(“model.h5”)

Note: To understand cross validation, refer to this link

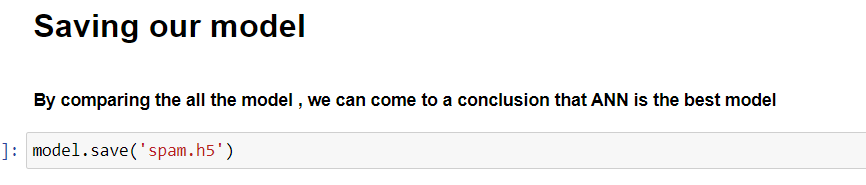


**Model Deployment**

In this milestone, we will see the model deployment

**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.



**Integrate With Web Framework**

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 server side script

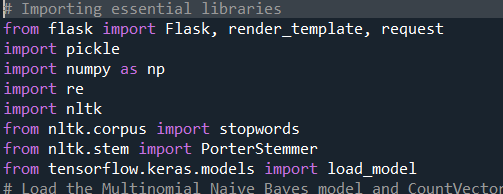
Run the web application

**Building Html Pages**

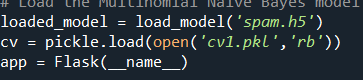
For this project create two HTML files namely

**Build Python Code**

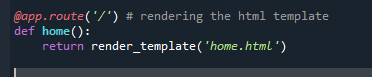
Import the libraries



Load the saved model. Importing the 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.



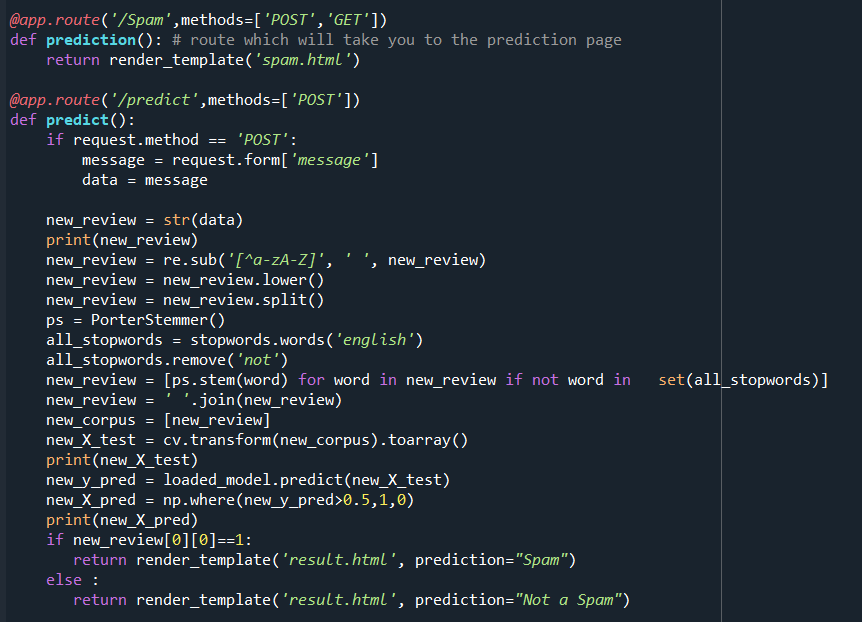
Render HTML page:



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

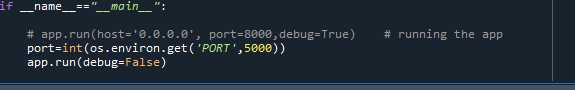
In the above example, ‘/’ URL is bound with the home.html function. Hence, when the home page of the web server is opened in the browser, the html page will be rendered. Whenever you enter the values from the html page the values can be retrieved using POST Method.

Retrieves the value from UI:



Here we are routing our app to predict() 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 be rendered to the text that we have mentioned in the submit.html page earlier.

Main Function:



**Run The Web Application**

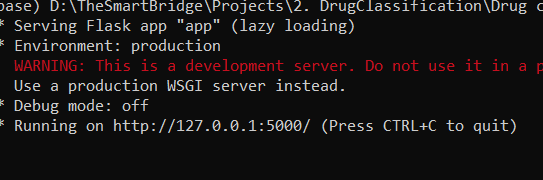
Open anaconda prompt from the start menu

Navigate to the folder where your python script is.

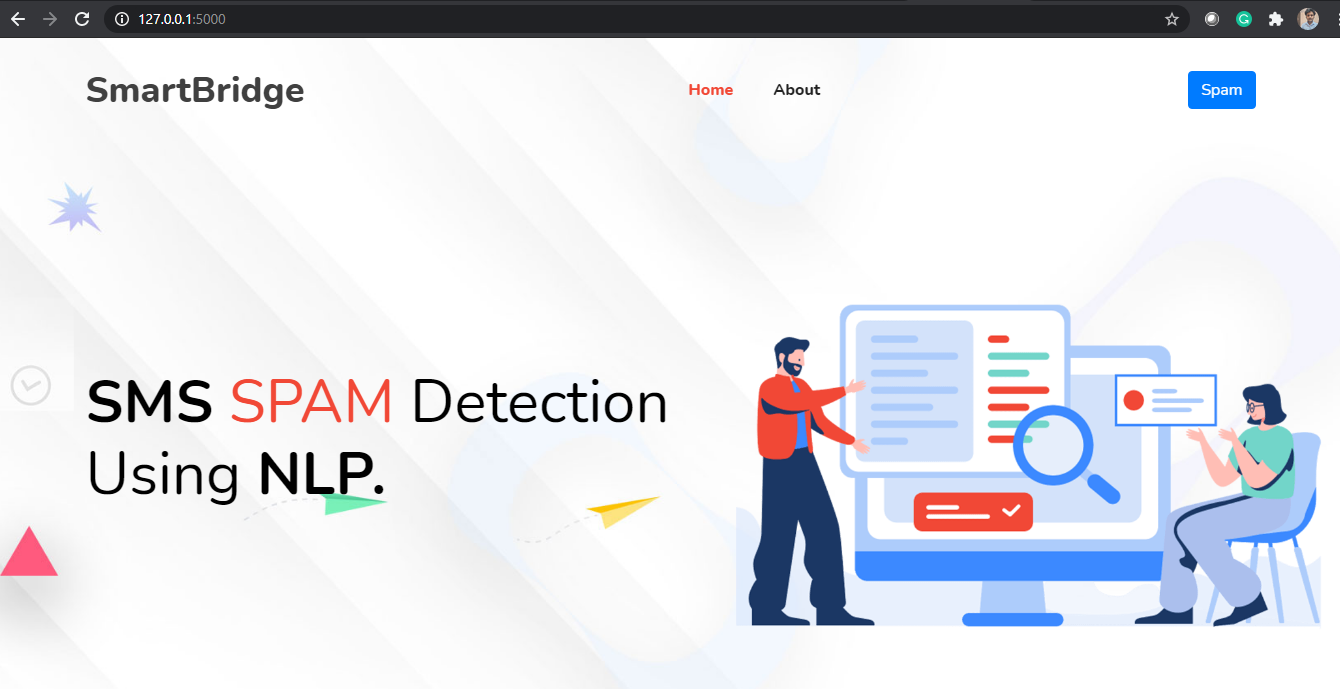
Now type “python app.py” command

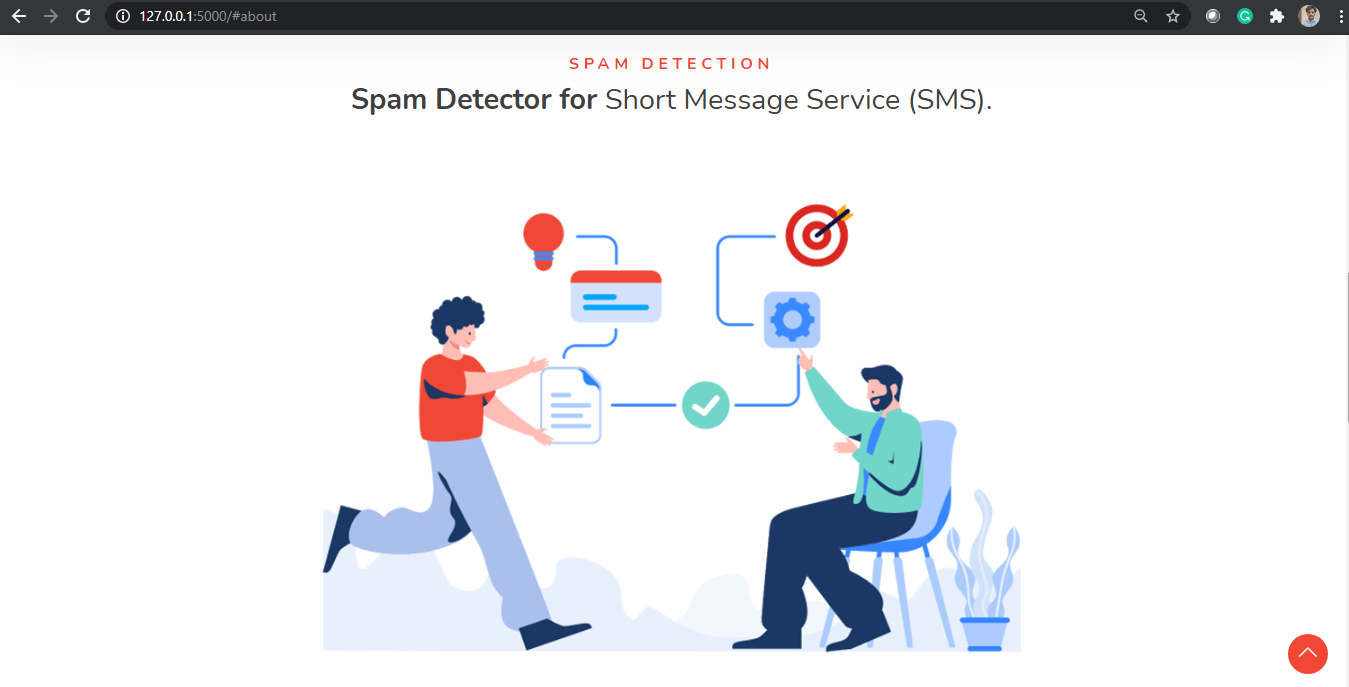
Navigate to the localhost where you can view your web page.

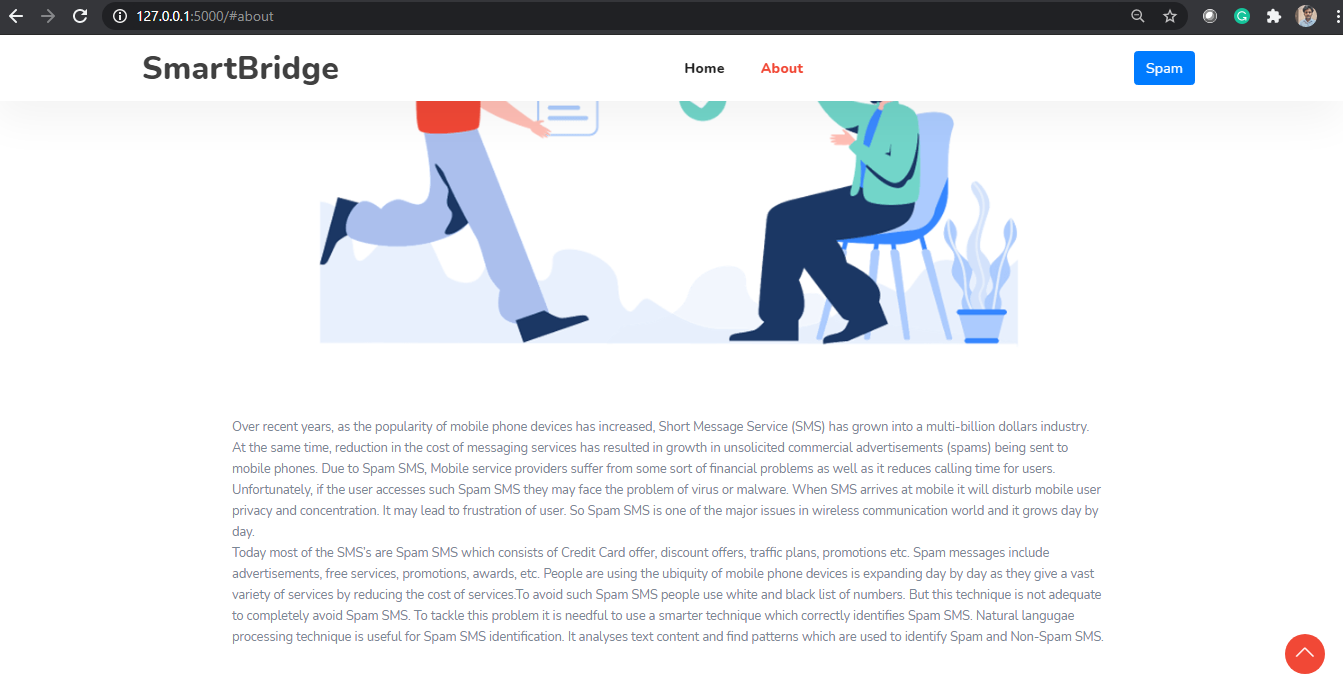
Click on the predict button from the top left corner, enter the inputs, click on the submit button, and see the result/prediction on the web.



This is the main page of Spam Detection , where you can know about the project and also from this page users can click onto the spam button and they will redirect onto the spam/ prediction page for providing the inputs.







**Project Demonstration & Documentation**

Project deliverables to be submitted along with other deliverables

**Record Explanation Video For Project End To End Solution**

Record explanation Video for project end to end solution

**Project Documentation-Step By Step Project Development Procedure**

Create document as per the template provided