CS-418 Introduction To Data Science Project Report

ON

Youtube comments spam classification

1.INTRODUCTION:

1.1. Project Overview:

YouTube is one of the largest and most popular video distribution platforms on the Internet and has more than 2 Billion users, it has more than **4 billion hours** worth of video viewers every month, and an estimated **500 hours** of video content are uploaded to YouTube every passing minute.

Many videos are uploaded on YouTube on a daily basis and a handsome amount of comments posted on those videos are spam, that is people trying to promote their channel or market their product. To tackle this problem, In this project we identify the YouTube video comments that belong to spam or ham(not spam).

1.2. Project aim:

Task: Binary classification problem to classify comments to spam or ham (not spam)

Input: Youtube comments

Output: Spam (or) Ham (Not spam)

This approach would aid in constructing an algorithm for blocking/removing inadmissible and undesirable comments from the view. In this project, NLP techniques are used to preprocess the data, before training the classification algorithm.

Primarily, the data set will be pre-processed and redundant data will be excluded. In subsequent phases, keywords (excluding common English words) are extracted using bag-of-words method and the frequency of usage is computed. Similarly, in the subsequent phases, essential features are derived by adding weight values to the data
using tf-idf feature extraction technique to form the distinct components of data used for further evaluation. On discrete data, Machine Learning Modelling will be applied. Logistic Regression, Random Forest Classifier, Support Vector Machine, Multi-layer Perceptron models are exercised on the refined training data to predict whether a comment is a Spam or Ham based on the patterns recognized. Lastly, testing data is used to validate the trained models. For this classification problem, evaluation metrics used are
Accuracy, precision, recall.

2. DATA DESCRIPTION

2.1 Dataset Overview:

The dataset is obtained from UCI Repository. It has five csv files composed of 1,956 tweet data extracted from five videos that were among the 10 most viewed on the collection period from 2015.The downloaded data folder has 5 data sets (csv files) :

1. Youtube01-Psy
2. Youtube02-KatyPerry

3. Youtube03-LMFAO
4. Youtube04-Eminem
5. Youtube05-Shakira

The dataset is formatted in a way where each row has a comment followed by a value marked as 1 or 0 for spam or not spam. The collection is composed by one CSV file per dataset, where each line has the following attributes:
COMMENT_ID, AUTHOR, DATE, CONTENT, TAG And Target class which has 0/1 values. Here is the **link** to the dataset.

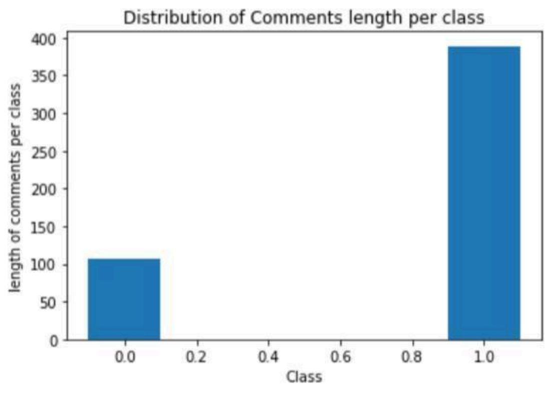
2.2. Required packages:

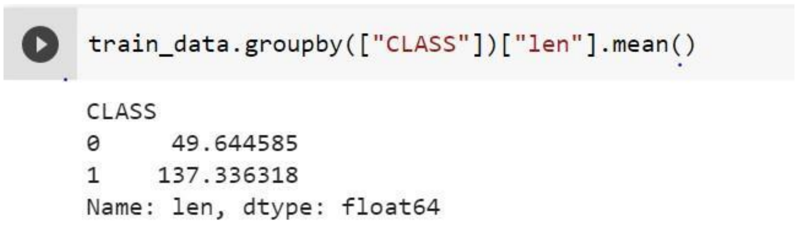
- Scikit Learn
- Numpy
- Pandas
- Matplotlib
- NLTK

2.3 Data cleaning and preprocessing:

Data has punctuations, encoding characters, URL links, contractions(don't, i'm ..), extra spaces, numeric digits and words containing digits, Emojis. Average length of comments is about 95 characters.

About 78% of the data that has length more than 95 are spam. Average length of spam comments is more than non-spam comments.





```
train_data.groupby(["CLASS"])["len"].mean()
```

```
CLASS
0     49.644585
1    137.336318
Name: len, dtype: float64
```

2.3.1 Dropping the irrelevant columns:

We need **CONTENT** and **CLASS** columns to perform spam classification tasks. Remaining attributes in the dataset can be removed.

- COMMENT_ID
- AUTHOR
- DATE
- len

These attributes are removed from the data set, we are classifying whether the comment is spam or not which has nothing to do with author,date or Comment_Id.

2.3.2. Expand contractions:

Contractions are words or combinations of words that are shortened by dropping letters and replacing them by an apostrophe.For better analysis of text data we need to expand contractions.For example:

we’re = we are

we’ve = we have

I’d = I would

Firstly, a computer does not know that contractions are abbreviations for a sequence of words. Therefore, a computer considers *we’re* and *we are* to be two completely different things and does not recognize that these two terms have the exact same meaning. Secondly, contractions increase the dimensionality of the document-term matrix. This increase in dimensionality makes computation more expensive.

2.3.3. Remove Html tags:

In our data set we have different html tags in the Content column,These HTML tags do not add any value to text data and only enable proper browser rendering. For example:

*<br />*http:/ www.gofundme.com/BishopsGraveMarker*<br />I* Or please just share it on your fb page

Here, for removing HTML tags, we will use BeautifulSoup, a python library for extracting and processing data with HTML and XML links.

2.3.3. Remove URL:

URLs (or Uniform Resource Locators) in a text are references to a location on the web, but do not provide any additional information. We thus remove these too using the library named re, which provides regular expression matching operations.

We take our sample text and analyze each word, removing words or strings starting with http.

2.3.4. Convert text to lowercase:

Converting all your data to lowercase helps in the process of preprocessing.

2.3.5. Remove numeric digits and words with digits:

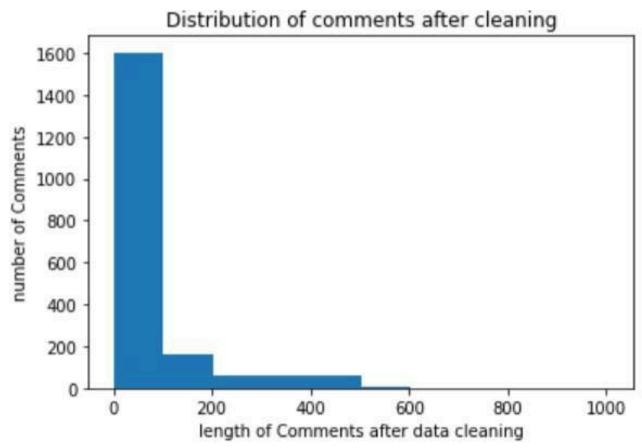
Data consists of some words attached with numeric values, we can remove them because it does not have any value in data.

2.3.6. Remove punctuations,encoding and Extra Spaces:

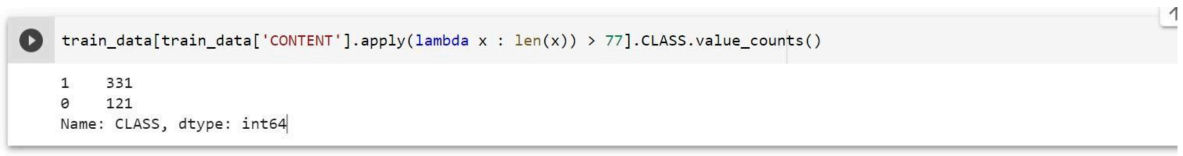
For text data analysis punctuations are not needed and Some comments have encoding \ufeff and \xao and emojis. This encoding part increases the dimensionality of the document-term matrix. Even the extra spaces have no values in the data set so removing it helps in data analysis.

2.4. After Data Cleaning:

Now after cleaning the data Average length of comments after cleaning the data is about 77.



```
Average lenght of comments after cleaning:- 74.8762781186094
```



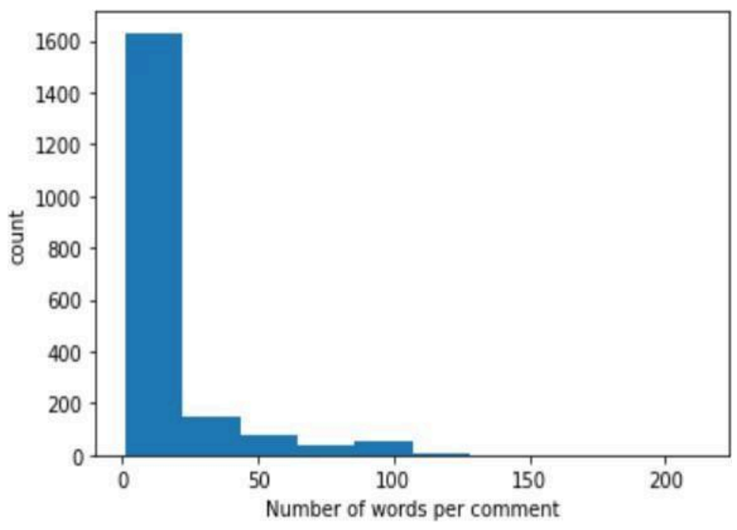
```
train_data[train_data['CONTENT'].apply(lambda x : len(x)) > 77].CLASS.value_counts()

1    331
0    121
Name: CLASS, dtype: int64
```

Almost 73% of data which has length more than average length are spam.

2.5. Feature Extraction:

2.5.1. Tokenization:

Tokens are the building blocks of Natural Language, the most common way of processing the raw text happens at the token level. Tokenization is the foremost step while modeling text data. Tokenization is performed on the corpus to obtain tokens. The following tokens are then used to prepare a vocabulary. Vocabulary refers to the set of unique tokens in the corpus.



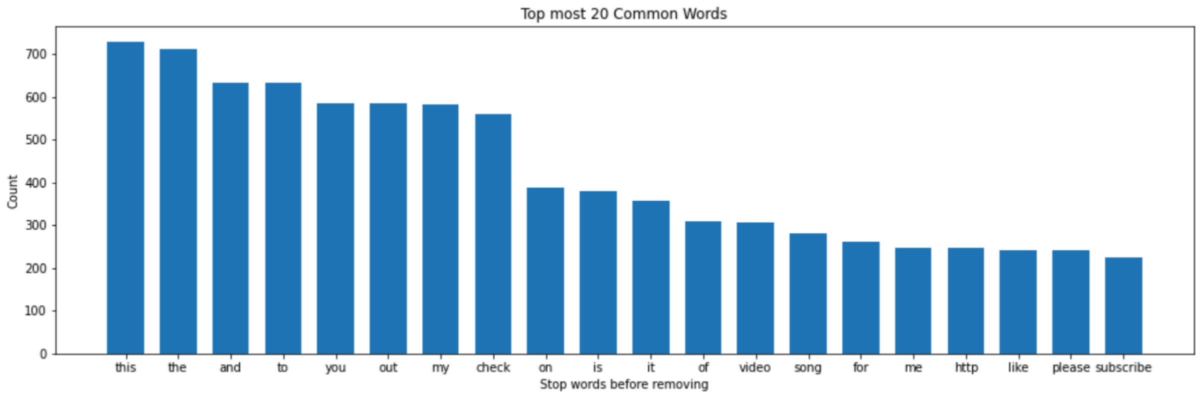
Distribution of tokens per comment, Most of the comments have length about 20-25.

2.5.2. Remove Stop Words:

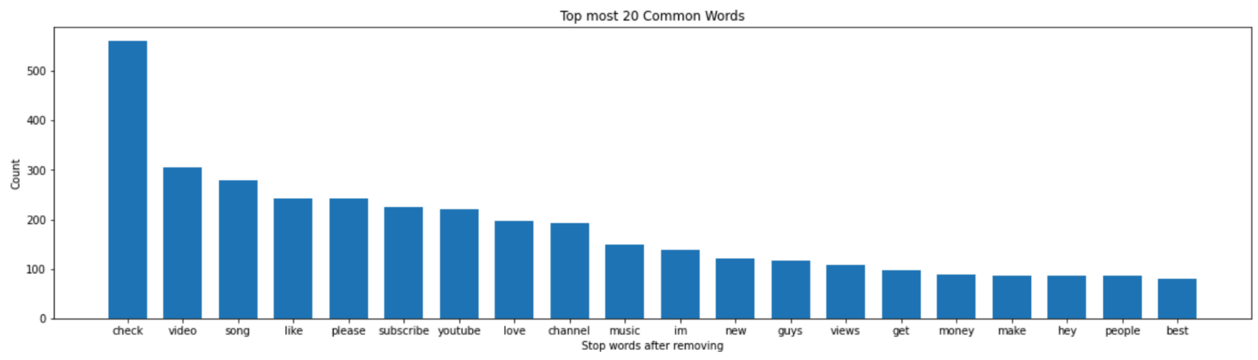
The words which are generally filtered out before processing a natural language are called stop words. These are actually the most common words in any language (like articles, prepositions, pronouns, conjunctions, etc) and do not add much information to the text.

Examples of a few stop words in English are “the”, “a”, “an”, “so”, “what”. Removal of stop words reduces the dataset size and thus reduces the training time due to the fewer number of tokens involved in the training. Removal of Stop Words removes low-level information from our text in order to give more focus to the important information.

Below plot is the distribution of the top 20 common words in the dataset vocabulary before removing stop words.



Below plot is the distribution of the top 20 common words in the dataset vocabulary after removing stop words.



2.5.3. Lemmatization:

Lemmatization is the method to normalize the text documents. In lemmatization, the word that is generated after chopping off the suffix is always meaningful and belongs to the dictionary, which means it does not produce any incorrect word. The word generated after lemmatization is also called a lemma. Below are a few examples of how lemmatization works.

am, are, is => be

car, cars, car's => car

Study, Studying, Studied => Study

2.5.4. TF-IDF Vectorizer:

After data cleaning, text data needed to be converted to numerical vectors as Machine learning models cannot train with text data. TF-IDF Vectorizer considers term frequency and the inverse document frequency while converting to numeric vector. The idea of tf-idf is to find the important words for the content of each document by decreasing the weight for commonly used words and increasing the weight for words that are not used very much in a collection or corpus of documents. Calculating tf-idf attempts to find the words that are important (i.e., common) in a text, but not *too* common.

2.6. Modelling:

Now that the data cleaning and feature extraction is done, We can train the models using different supervised machine learning algorithms.

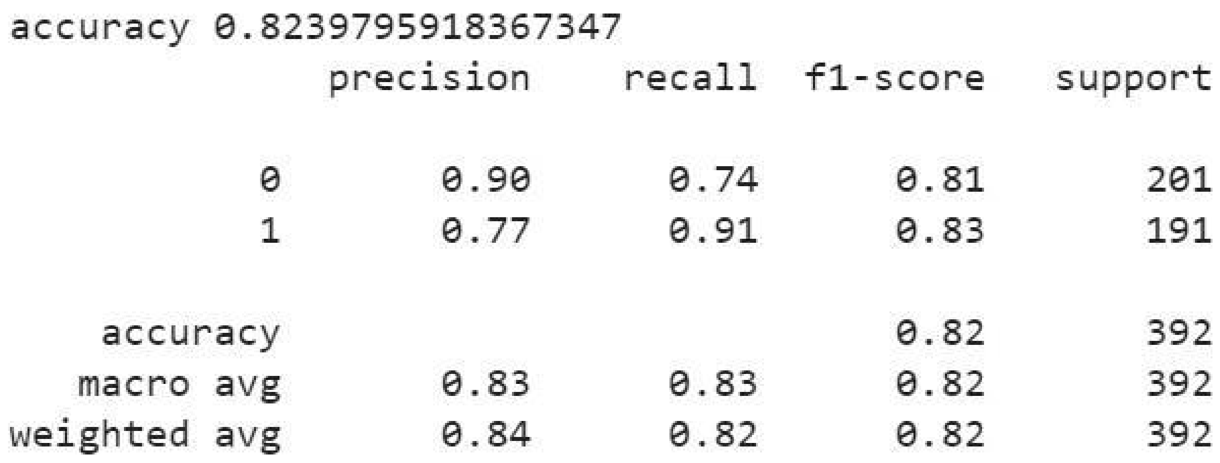
Models used are:

- Logistic Regression
- Naive Bayes
- Support Vector Machine
- Random Forest

2.6.1. Logistic Regression:

Logistic Regression is used here as it is generally used to predict the happening of an event with respect to the dependent factors. Logic Regression uses a Logistic function that models quantities' growth rate over the function of other quantities.

We used an 80-20 train-test split for this model.
The hyper parameters used are n_jobs=1, C=1e5, max_iter = 200. Performance metrics of the model are as below
Accuracy obtained : 82%



```
accuracy 0.8239795918367347
              precision    recall  f1-score   support

           0       0.90      0.74      0.81       201
           1       0.77      0.91      0.83       191

    accuracy                           0.82       392
   macro avg       0.83      0.83      0.82       392
weighted avg       0.84      0.82      0.82       392
```

2.6.2. Naive Bayes:

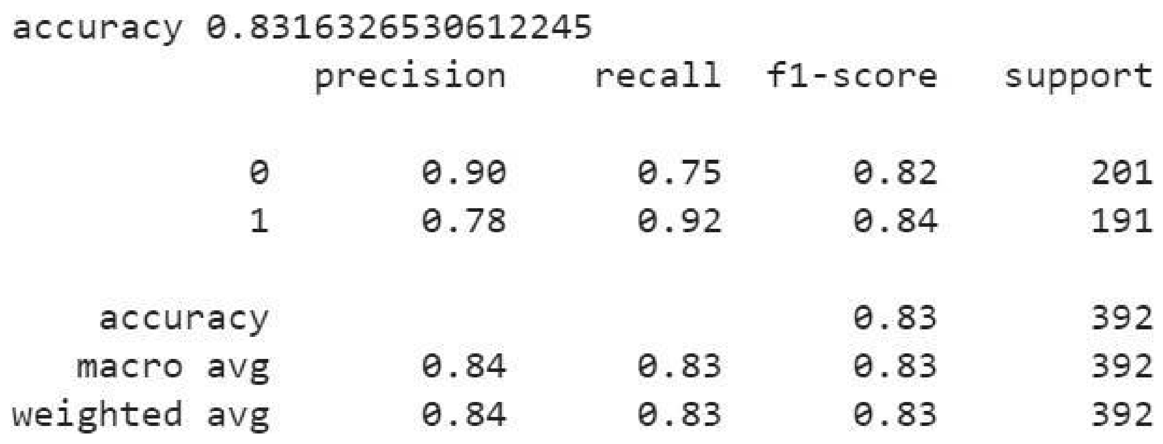
A Naive Bayes classifier is a probabilistic model that is used for classification tasks. The crux of the classifier is based on bayes theorm.

$$P(A|B) = P(B|A) * P(A) / P(B)$$

80% train data 20% test data split data used to train naive bayes model. Multinomial Naive Bayes classifier is used.

Performance metrics of the model are as below

Accuracy obtained: 83%



```
accuracy 0.8316326530612245
              precision    recall  f1-score   support

           0       0.90      0.75      0.82       201
           1       0.78      0.92      0.84       191

    accuracy                           0.83       392
   macro avg       0.84      0.83      0.83       392
weighted avg       0.84      0.83      0.83       392
```

2.6.3. Support Vector Machine:

Support Vector Machine (SVM) is a Linear Supervised model for classification and regression. SVM algorithm works on the principle of separating classes of data such that each class consists of similar elements and distinct elements are placed in the different classes by analyzing the features and comparing it with different samples in the training set. Separating classes of data is achieved by margins, hyper planes and support vectors. In the context of SVM, margin correlates to area between the two classes of data. There might be some data that could be present between the areas of separation and are called outliers, those belonging to either of the classes. To estimate the proximity of outliers with respect to either of the classes, a hyperplane is drawn in the area of margin. Larger the margin area, higher the probability of better partition of classes and accordingly better likelihood of classification.

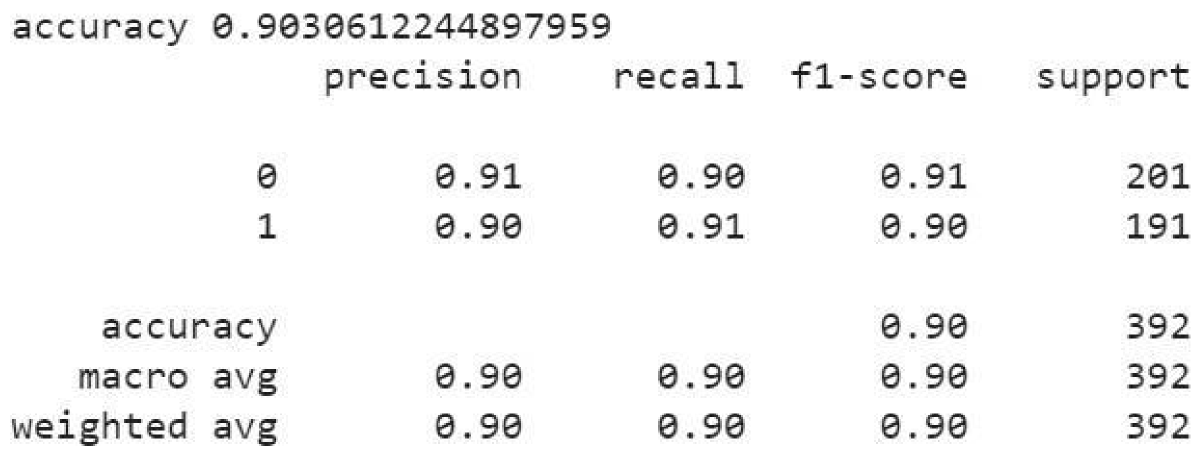
80% train data 20% test data split data used to train Support Vector

Machine. Linear kernel and Stochastic Gradient Descent Classifier are used.

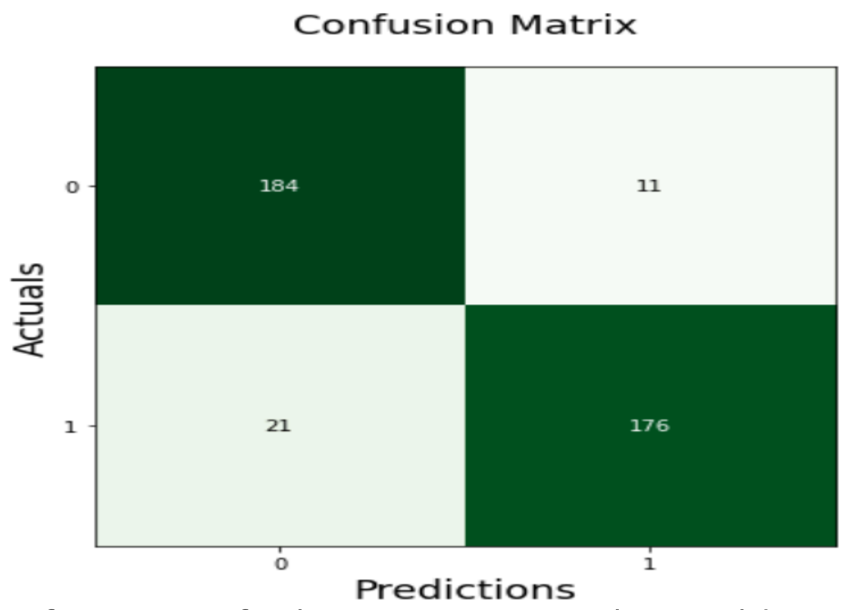
Hyper parameters used are 5 iterations with Learning_Rate 0.003.

Performance metrics of the model are as below

Accuracy obtained: 90%



```
accuracy 0.9030612244897959
              precision    recall  f1-score   support

           0       0.91      0.90      0.91       201
           1       0.90      0.91      0.90       191

    accuracy                           0.90       392
   macro avg       0.90      0.90      0.90       392
weighted avg       0.90      0.90      0.90       392
```



Above is the confusion matrix for the support vector machine model.

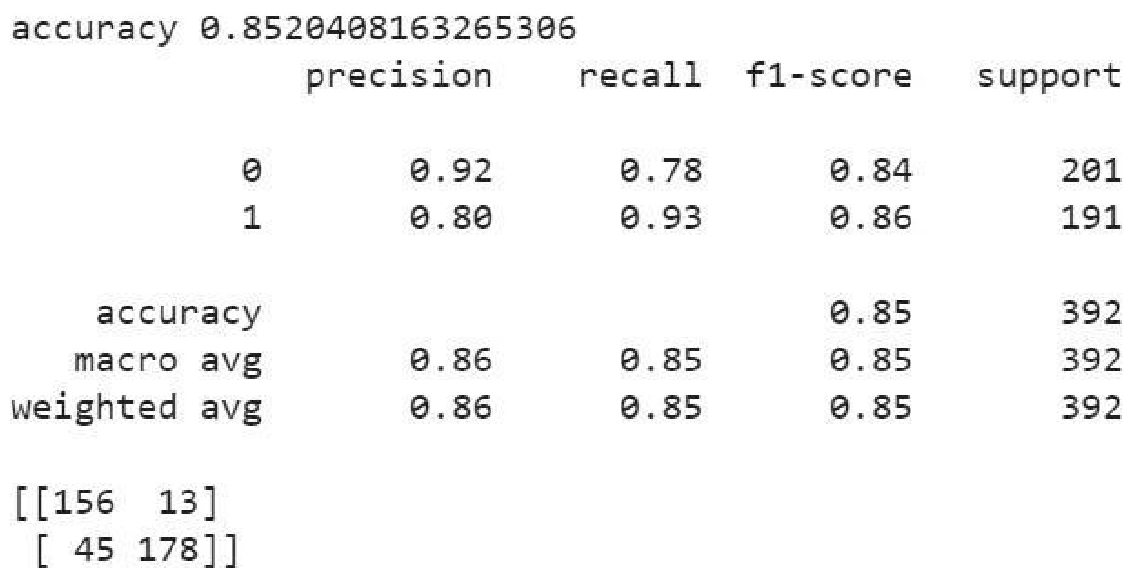
2.6.4. Random Forest:

Random Forest Classifier (RMF) is an ensemble of decision trees built on the basis of a supervised algorithm. Usually, overfitting is the issue in the machine learning models as the models fit the training data too close to the set of data points. This issue happens when the model understands the noise too closely. Usually, to prevent overfitting, regularization is added or the nodes in the hidden layers are diminished. But RMF does it by default as the group of decision trees run altogether.
80% train data 20% test data split data used to trainRandom Forest.

Linear kernel and Stochastic Gradient Descent Classifier are used.

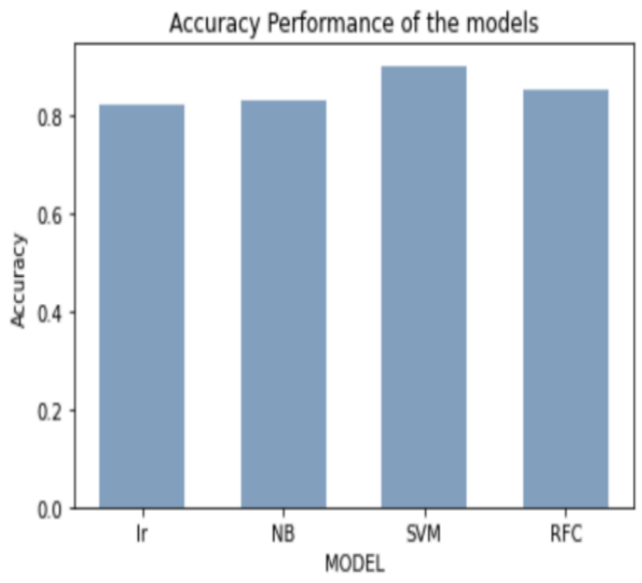
Hyper parameters used are ‘criterion’: [‘gini’, ‘entropy’], ‘max_depth’: [5, 10, 20].

Performance metrics of the model are as below.
Accuracy obtained: 86%



```
accuracy 0.8520408163265306
              precision    recall  f1-score   support

           0       0.92      0.78      0.84       201
           1       0.80      0.93      0.86       191

    accuracy                           0.85       392
   macro avg       0.86      0.85      0.85       392
weighted avg       0.86      0.85      0.85       392

[[156  13]
 [ 45 178]]
```

2.7. Model Results:



Model	Accuracy	Precision	Recall
Logistic Regression	82	91	76
Naive Bayes	83	91	77
Support Vector Machine	90	90	89
Random Forest	85	93	79

3. Conclusion:

Support vector machine performed well compared to other models. SVM obtained not only good accuracy but also good recall and precision values. Random Forest model performed well after SVM model. Precision is highest for Random Forest among all the models. This project can be further improved by applying Deep Learning models. The Future Scope for this project can be deploying the model.