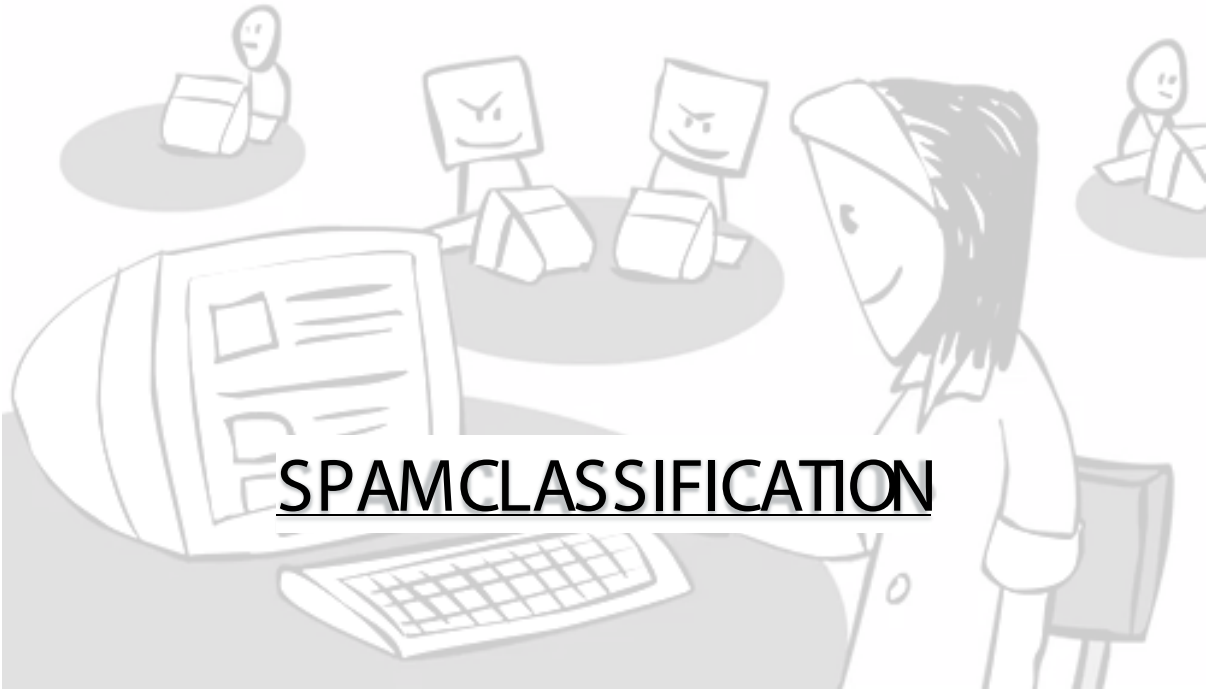
FLIP ROBO



SPAMCLASSIFICATION

SUBMITTED BY

Abhijit Sarkar

ACKNOWLEDGMENT

I would like to express my gratitude for the opportunity and would like to thank Flip Robo Technologies for giving me an opportunity to work on this project of SPAM CLASSIFICATION. While going through this project I could somewhere find how text can be converted to vectors, and those vectors can be used for spam classification. More I knew that how these spam messages are sent to a large group of recipients without their prior consent, typically advertising for goods and services or business opportunities. In recent days the percentage of spam messages have gone very high. Even many frauds have been reported to be police and many got even highlighted in newspapers. I am very grateful to DATA TRAINED team for providing me the adequate Trainings which actually helped me a lot to completion of this project. I took help from Mr. Mohd Kashif and the document links and study materials provided during project completion was very helpful.

During the completion of the project, I found various issues working with NLP and I overcome those problem with the help of Java T point, You Tube Videos of Mr. Nair, Kaggle and Medium.

INTRODUCTION

1. BUSINESS PROBLEM FRAMING

Spam messages are messages sent to a large group of recipients without their prior consent, typically advertising for goods and services or business opportunities. In the recent period, the percentage of scam messages amongst spam have increased sharply. Scam messages typically trick people into giving away money or personal details by offering an attractive or false deal. Based on the statistics from the Singapore Police Force, from January till June 2020, the amount cheated through scams have increased by more than S$8 million.

A spam message classification is a step towards building a tool for scam message identification and early scam detection.

2. CONCEPTUAL BACKGROUND OF THE DOMAIN PROBLEM

Spam Detector is used to detect unwanted, malicious and virus infected texts and helps to separate them from the non-spam texts. It uses a binary type of classification containing the labels such as **HAM** (which is not a spam) and a **SPAM (**which is a spam**)**. We can this type of Application, These Application of can be seen in Google Mail (GMAIL) where it segregates the spam emails and non-spam emails in order to prevent them from getting into the user's inbox.

Those concepts can be used to detect the SMS and classify them on the basis of spam and non-spam.

3. REVIEW OF LITERATURE

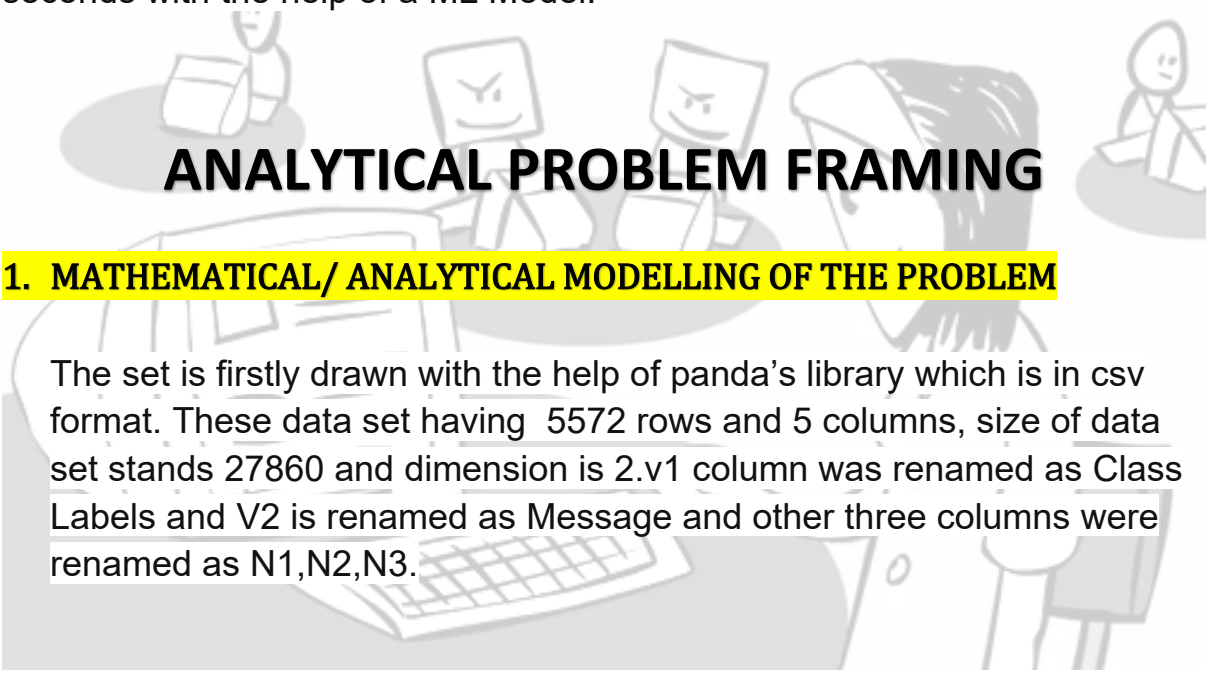
The files contain one message per line. Each line is composed by four columns, in which v1 contains the class label {HAM and SPAM) and v2 column contains

the raw text messages received from users. v3, v4 are rows having message texts. This corpus has been collected from free or free for research sources at the Internet.

4. MOTIVATION FOR THE PROBLEM UNDERTAKEN

Spam classifier machine learning model is needed to segregate the mails, SMS and its important because manually checking Mails and SMS is a boring and more time-consuming task. Based on manually selection and rejection a SMS and Mail can't be much accurate.

We can save huge amount of Time and do those boring task in couple of seconds with the help of a ML Model.



ANALYTICAL PROBLEM FRAMING

1. MATHEMATICAL/ ANALYTICAL MODELLING OF THE PROBLEM

The set is firstly drawn with the help of panda's library which is in csv format. These data set having 5572 rows and 5 columns, size of data set stands 27860 and dimension is 2.v1 column was renamed as Class Labels and V2 is renamed as Message and other three columns were renamed as N1,N2,N3.

- The unique Values of the data set was checked.
- Null values were checked along with the Percentage calculation was drawn from the Data Set in the form of a Data Frame, Three columns shows almost 99 percentage of the data were missing, However the missing content was more than 90 percent, Hence those columns were dropped.

- Duplicated values were checked, found 403 contents to be duplicated and those columns were dropped.

- Finally, after Imputation the Visualization was created and Next step was taken towards data Analysis.

- Length of the Total characters, words and sentences were drawn to find the relation between them.

- Various visualization was performed on the same.

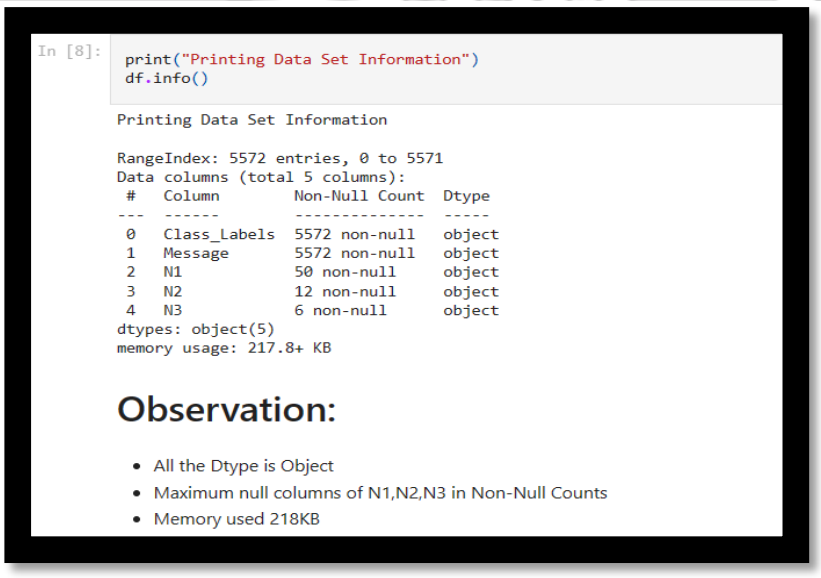
2. DATA SOURCES AND THEIR FORMATS

This Data Set was provided by Flip Robo Technology to learn various aspects of the NLTK library. The data set contains

i. A collection of 5573 rows SMS spam messages was manually extracted from the Grumble text Web site. This is a UK forum in which cell phone users make public claims about SMS spam messages, most of them without reporting the very spam message received.

ii. A subset of 3,375 SMS randomly chosen ham messages of the NUS SMS Corpus (NSC), which is a dataset of about 10,000 legitimate messages collected for research at the Department of Computer Science at the National University of Singapore.

iii. This Data Sets contains 05 objects and Memory consumed is 218KB



3. DATA PRE-PROCESSING DONE

Data Pre-processing

- Loading Data set, Checking and imputing null values, Checking Duplicated rows in columns and eliminating the rows those were duplicated.
- Using Encoding Methods.
- The Length of the Total Characters, words and sentences are captured and various measures were drawn in the form of a number.
- Descriptive analysis was performed.
- Correlation of the data set were checked and correlation were checked among independent variable were checked.

Text Pre- Processing

- Texts were drawn lower using string name. lower function.
- Tokenization was done (sentences were drawn into words)
- Stop Words were used ("English")
- Texts Stemming were done as Stemming works better on spam classification.
- Then corpus was joint in the form of a string join.

In [54]:

```
def transform(text):
  text=text.lower() # make in lower case
  text=nltk.word_tokenize(text) # tokenized each word

  corpus=[]
  for i in text:
    if i.isalnum():# if text is alpha-numerical
      corpus.append(i) # append text

  new_corpus=corpus.copy() # making a copy
  corpus.clear() # clearing corpus

  for i in new_corpus: # from new corpus

 # using stop words of english language and all punctuations
    if i not in stopwords.words("english") and i not in string.punctuation:
      corpus.append(i) # appending

  new_corpus=corpus.copy() # making a copy
  corpus.clear() #clearing the copy

  ps=PorterStemmer() #importing Porter Stemmer

  for i in new_corpus:
    corpus.append(ps.stem(i)) # used stemming process

  return " ".join(corpus)
```

4. DATA INPUTS- LOGIC- OUTPUT RELATIONSHIPS

- There is 04 independent variable and 01 dependent variable.
- I have used count plot, cat plot, bar plot, box plot, distplot for univariate analysis.
- I have used count plot and cat plot to check the relation between independent and dependent variable.
- We used person correlation to check the relation between variables.
- We have totally used univariate, bivariate and multivariate graph to determine the relation among variables.
- We used Word Clouds to check the frequent used spam words and ham words to draw the relations among the independent variable and dependent variable.

5. STATE THE SET OF ASSUMPTIONS (IF ANY) RELATED TO THE PROBLEM UNDER CONSIDERATION

1. Naïve Bayes classifier

It is a supervised machine learning algorithm where words probabilities play the main rule here. If some words occur often in spam but not in ham, then this incoming e-mail is probably spam. Naïve bayes classifier technique has become a very popular method in mail filtering software.

2. Artificial Neural Networks classifier:

An artificial neural network (ANN), also called simply a "Neural Network" (NN), is a computational model based on biological neural networks. It consists of an interconnected collection of artificial neurons. An artificial neural network is an adaptive system that changes its structure based on information that flows through the artificial network during a learning phase.

6. HARDWARE AND SOFTWARE REQUIREMENTS AND TOOLS USED

Hardware:

SOFTWARE USED:

- ***I Used Jupiter Note Book.***
- ***Microsoft Office 2020***
- ***Windows 11 OS***

Library used: To run the program and to build the model we need some basic libraries as follows:

- ***NumPy***
- ***Pandas***
- ***Seaborn***
- ***Matplotlib***
- ***SciPy***
- ***Sklearn***
- ***Pickle***
- ***Imbalance Learn***
- ***NLTK***

1) ***import pandas as pd:***
   Pandas are a popular Python-based data analysis toolkit which can be imported using import pandas as pd. It presents a diverse range of utilities, ranging from parsing multiple file formats to converting an entire data table into a NumPy matrix array. This makes pandas a trusted ally in data science and machine learning.
2) ***import NumPy as np:***
   NumPy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more.

3) ***import seaborn as sns:***
   Seaborn is a data visualization library built on top of matplotlib and closely integrated with pandas' data structures in Python. Visualization is the central part of Seaborn which helps in exploration and understanding of data.

4) ***Import matplotlib.pyplot as plt:***

matplotlib.pyplot is a collection of functions that make matplotlib work like MATLAB. Each pyplot function makes some change to a figure: e.g., creates a figure, creates a plotting area in a figure, plots some lines in a plotting area, decorates the plot with labels, etc.

5. scipy:

SciPy is a free and open-source Python library used for scientific computing and technical computing. SciPy contains modules for optimization, linear algebra, integration, interpolation, special functions, FFT, signal and image processing, ODE solvers and other tasks common in science and engineering.

6. Sklearn

Scikit-learn is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support-vector machines.

7. Pickle

"Pickling" is the process whereby a Python object hierarchy is converted into a byte stream, and "unpickling" is the inverse operation, whereby a byte stream (from a binary file or bytes-like object) is converted back into an object hierarchy.

8. IMB learn (SMOTE)

One approach to addressing imbalanced datasets is to oversample the minority class. The simplest approach involves duplicating examples in the minority class, although these examples don't add any new information to the model. Instead, new examples can be synthesized from the existing examples. This is a type of data augmentation for the minority class and is referred to as the Synthetic Minority Oversampling Technique, or SMOTE for short.

9.NLTK

The Natural Language Toolkit, or more commonly NLTK, is a suite of libraries and programs for symbolic and statistical natural language processing for English written in the Python programming language.

Model/s Development and Evaluation.

- ## Identification of possible problem-solving approaches (methods)

  i. Length of the sentences of each line were drawn into a numeric value in separate column named Total Sentences.

  ii. Length of the words of each line is drawn into a numeric column named Total Words.

iii. Length of the total characters were drawn into numeric values named Total Characters.

iv. Most frequent words were checked using word clouds and collection library.

➢ Testing of Identified Approaches (Algorithms)

adaptive system that changes its structure based on information that flows through the artificial network during a learning phase.

1. We have used Naïve Bayes Models:
   - *gnb=GaussianNB()*
   - *mnb=MultinomialNB()*
   - *bnb=BernoulliNB()*

2. We have used Other Models:
   - *lg=LogisticRegression()*
   - *dtc=DecisionTreeClassifier()*
   - *knn=KNeighborsClassifier()*
   - *svc=SVC()*
   - *rfc=RandomForestClassifier()*
   - *etc=ExtraTreesClassifier()*
   - *gbc=GradientBoostingClassifier()*
   - *sgd=SGDClassifier()*
   - *mlp=MLPClassifier()*

➢ Run and evaluate selected models

1.

```
*********************************
 GaussianNB()
 *********************************
Training Score 0.924788391777509 And Precision Score 47.43589743589743
Accuracy Training Score = 0.924788391777509  Accuracy Test Score = 0.8646034816247582

Training Confusion_Matrix
 [[3299  311]
 [   0  525]] Testing Confusion_Matrix
 [[783 123]
 [ 17 111]]
Classification Report
              precision    recall  f1-score   support

           0       0.98      0.86      0.92       906
           1       0.47      0.87      0.61       128

    accuracy                           0.86      1034
   macro avg       0.73      0.87      0.77      1034
weighted avg       0.92      0.86      0.88      1034

 * * * * *
```

2.

```
*******************************
 MultinomialNB()
 *******************************
Training Score 0.992261185006046 And Precision Score 86.52482269503547
Accuracy Training Score = 0.992261185006046  Accuracy Test Score = 0.9758220502901354

Training Confusion_Matrix
 [[3599   11]
 [  21  504]] Testing Confusion_Matrix
 [[887  19]
 [  6 122]]
Classification Report
              precision    recall  f1-score   support

           0       0.99      0.98      0.99       906
           1       0.87      0.95      0.91       128

    accuracy                           0.98      1034
   macro avg       0.93      0.97      0.95      1034
weighted avg       0.98      0.98      0.98      1034

 * * * * *
```

3.

```
*******************************
 BernoulliNB()
 *******************************
Training Score 0.97726723095526 And Precision Score 94.5945945945946
Accuracy Training Score = 0.97726723095526  Accuracy Test Score = 0.971953578336557

Training Confusion_Matrix
 [[3607    3]
 [  91  434]] Testing Confusion_Matrix
 [[900   6]
 [ 23 105]]
Classification Report
              precision    recall  f1-score   support

           0       0.98      0.99      0.98       906
           1       0.95      0.82      0.88       128

    accuracy                           0.97      1034
   macro avg       0.96      0.91      0.93      1034
weighted avg       0.97      0.97      0.97      1034

 * * * * *
```

4.

```
*******************************
 LogisticRegression()
 *******************************
Training Score 0.9949214026602177 And Precision Score 100.0
Accuracy Training Score = 0.9949214026602177  Accuracy Test Score = 0.9806576402321083

Training Confusion_Matrix
 [[3609    1]
 [  20  505]] Testing Confusion_Matrix
 [[906   0]
 [ 20 108]]
Classification Report
              precision    recall  f1-score   support

           0       0.98      1.00      0.99       906
           1       1.00      0.84      0.92       128

    accuracy                           0.98      1034
   macro avg       0.99      0.92      0.95      1034
weighted avg       0.98      0.98      0.98      1034

 * * * * *
```

5.

```
*******************************
 DecisionTreeClassifier()
 *******************************
Training Score 1.0 And Precision Score 88.18181818181819
Accuracy Training Score = 1.0  Accuracy Test Score = 0.9574468085106383

Training Confusion_Matrix
 [[3610    0]
 [   0  525]] Testing Confusion_Matrix
 [[893  13]
 [ 31  97]]
Classification Report
              precision    recall  f1-score   support

           0       0.97      0.99      0.98       906
           1       0.88      0.76      0.82       128

    accuracy                           0.96      1034
   macro avg       0.92      0.87      0.90      1034
weighted avg       0.96      0.96      0.96      1034
```

6.

```
*******************************
 KNeighborsClassifier()
 *******************************
Training Score 0.9238210399032648 And Precision Score 100.0
Accuracy Training Score = 0.9238210399032648  Accuracy Test Score = 0.9177949709864603

Training Confusion_Matrix
 [[3610    0]
 [ 315  210]] Testing Confusion_Matrix
 [[906   0]
 [ 85  43]]
Classification Report
              precision    recall  f1-score   support

           0       0.91      1.00      0.96       906
           1       1.00      0.34      0.50       128

    accuracy                           0.92      1034
   macro avg       0.96      0.67      0.73      1034
weighted avg       0.92      0.92      0.90      1034

 * * * * *
```

7.

```
*******************************
 SVC()
 *******************************
Training Score 0.9949214026602177 And Precision Score 100.0
Accuracy Training Score = 0.9949214026602177  Accuracy Test Score = 0.9680851063829787

Training Confusion_Matrix
 [[3610    0]
 [  21  504]] Testing Confusion_Matrix
 [[906   0]
 [ 33  95]]
Classification Report
              precision    recall  f1-score   support

           0       0.96      1.00      0.98       906
           1       1.00      0.74      0.85       128

    accuracy                           0.97      1034
   macro avg       0.98      0.87      0.92      1034
weighted avg       0.97      0.97      0.97      1034
```

08.

```
*****************************
 RandomForestClassifier()
 *****************************
Training Score 1.0 And Precision Score 98.9795918367347
Accuracy Training Score = 1.0  Accuracy Test Score = 0.9690522243713733

Training Confusion_Matrix
 [[3610    0]
 [   0  525]] Testing Confusion_Matrix
 [[905   1]
 [ 31  97]]
Classification Report
              precision    recall  f1-score   support

           0       0.97      1.00      0.98       906
           1       0.99      0.76      0.86       128

    accuracy                           0.97      1034
   macro avg       0.98      0.88      0.92      1034
weighted avg       0.97      0.97      0.97      1034

 * * * * *
```

09.

```
*****************************
 ExtraTreesClassifier()
 *****************************
Training Score 1.0 And Precision Score 99.0
Accuracy Training Score = 1.0  Accuracy Test Score = 0.9709864603481625

Training Confusion_Matrix
 [[3610    0]
 [   0  525]] Testing Confusion_Matrix
 [[905   1]
 [ 29  99]]
Classification Report
              precision    recall  f1-score   support

           0       0.97      1.00      0.98       906
           1       0.99      0.77      0.87       128

    accuracy                           0.97      1034
   macro avg       0.98      0.89      0.93      1034
weighted avg       0.97      0.97      0.97      1034
```

10.

```
*****************************
 GradientBoostingClassifier()
 *****************************
Training Score 0.9743651753325272 And Precision Score 96.875
Accuracy Training Score = 0.9743651753325272  Accuracy Test Score = 0.9632495164410058

Training Confusion_Matrix
 [[3609    1]
 [ 105  420]] Testing Confusion_Matrix
 [[903   3]
 [ 35  93]]
Classification Report
              precision    recall  f1-score   support

           0       0.96      1.00      0.98       906
           1       0.97      0.73      0.83       128

    accuracy                           0.96      1034
   macro avg       0.97      0.86      0.90      1034
weighted avg       0.96      0.96      0.96      1034
```

11.

```
*******************************
 SGDClassifier()
 *******************************
Training Score 0.999758162031439 And Precision Score 99.12280701754386
Accuracy Training Score = 0.999758162031439  Accuracy Test Score = 0.9845261121856866

Training Confusion_Matrix
 [[3610    0]
 [   1  524]] Testing Confusion_Matrix
 [[905   1]
 [ 15 113]]
Classification Report
              precision    recall  f1-score   support

           0       0.98      1.00      0.99       906
           1       0.99      0.88      0.93       128

    accuracy                           0.98      1034
   macro avg       0.99      0.94      0.96      1034
weighted avg       0.98      0.98      0.98      1034
```

12.

```
*******************************
 MLPClassifier()
 *******************************
Training Score 1.0 And Precision Score 99.0909090909091
Accuracy Training Score = 1.0  Accuracy Test Score = 0.9806576402321083

Training Confusion_Matrix
 [[3610    0]
 [   0  525]] Testing Confusion_Matrix
 [[905   1]
 [ 19 109]]
Classification Report
              precision    recall  f1-score   support

           0       0.98      1.00      0.99       906
           1       0.99      0.85      0.92       128

    accuracy                           0.98      1034
   macro avg       0.99      0.93      0.95      1034
weighted avg       0.98      0.98      0.98      1034
```

➢ Key Metrics for success in solving problem under consideration

- Precision Score
- Model Training Score
- Accuracy Score
- Confusion Metrics
- Classification Report

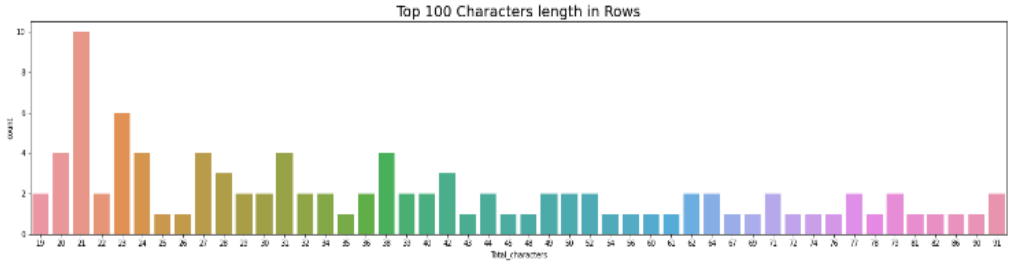
➢ Visualizations

Univariate Analysis on the data

In [28]:

```
plt.figure(figsize=(25,5))
sns.countplot(x=df["Total_characters"].value_counts()[:100],data=df)
plt.title("Top 100 Characters length in Rows",Fontsize=20)
print("Minimum Character :",df['Total_characters'].min(),"\nMaximum Characters :",df['Tota
```

```
Minimum Character : 2
Maximum Characters : 910
```

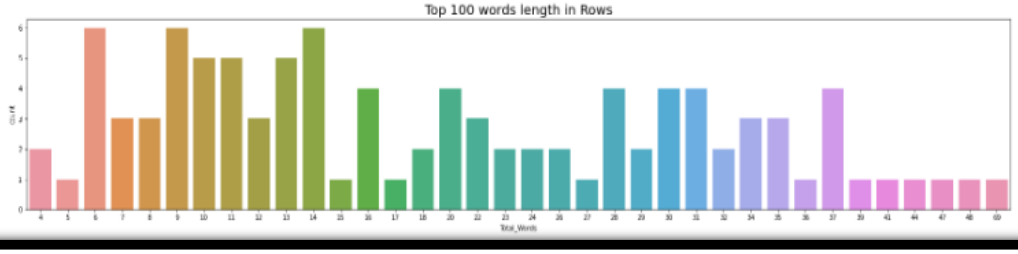


Shows Total

In [29]:

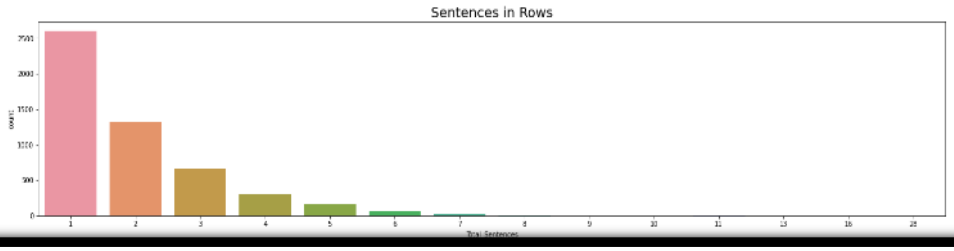
```
plt.figure(figsize=(28,5))
sns.countplot(x=df["Total_Words"][:100],data=df)
plt.title("Top 100 words length in Rows",Fontsize=20)
print("Minimum Words :",df['Total_Words'].min(),"\nMaximum Words :",df['Total_Words'].max(
```

```
Minimum Words : 1
Maximum Words : 220
```



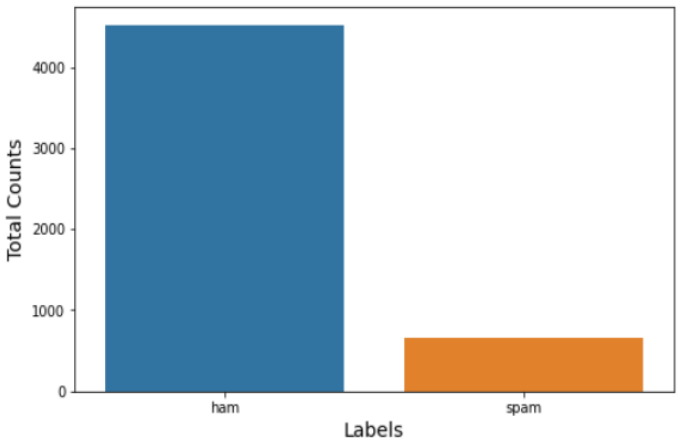
```
In [30]: plt.figure(figsize=(25,5))
         sns.countplot(x=df['Total_Sentences'],data=df)
         plt.title("Sentences in Rows",Fontsize=20)
         print("Minimum Sentences :",df['Total_Sentences'].min(),"\nMaximum Sentences :",df['Total_
```

```
Minimum Sentences : 1
Maximum Sentences : 28
```

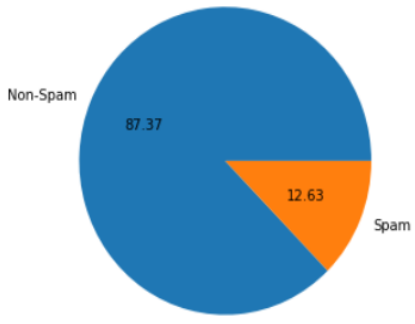


Checking Ham and Spam

```
ham     4516
spam     653
Name: Class_Labels, dtype: int64
```



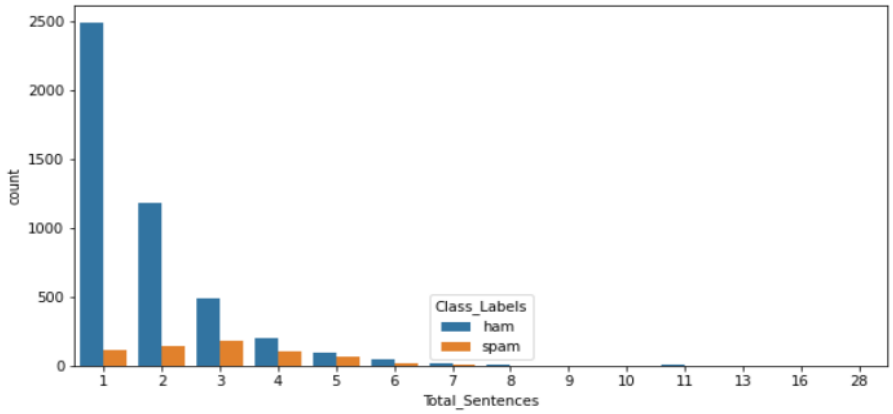

```
plt.figure(figsize=(6,5))
plt.pie(df['Class_Labels'].value_counts(),labels=['Non-Spam','Spam'],autopct="%0.02f")
plt.show()
```



Bi-Variate Analysis

In [32]:
```
plt.figure(figsize=(10,5))
sns.countplot(x=df['Total_Sentences'],hue=df['Class_Labels'],data=df)
```

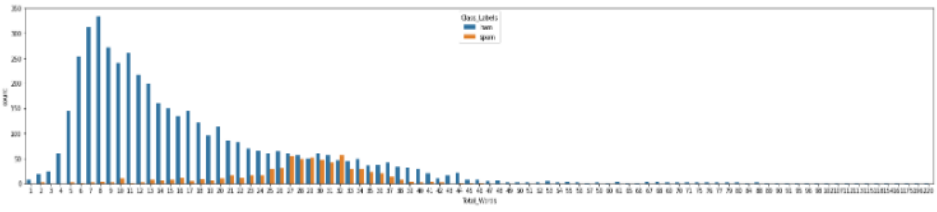
Out[32]:



Observations

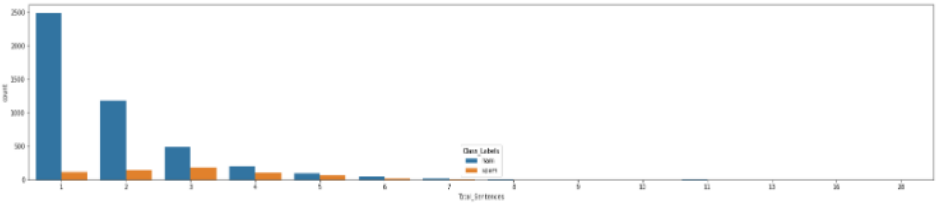
- Possibility of Spam increases with increase in sentences

In [35]:
```
plt.figure(figsize=(28,5))
sns.countplot(x=df['Total_Words'],hue=df['Class_Labels'],data=df)
plt.show()
```



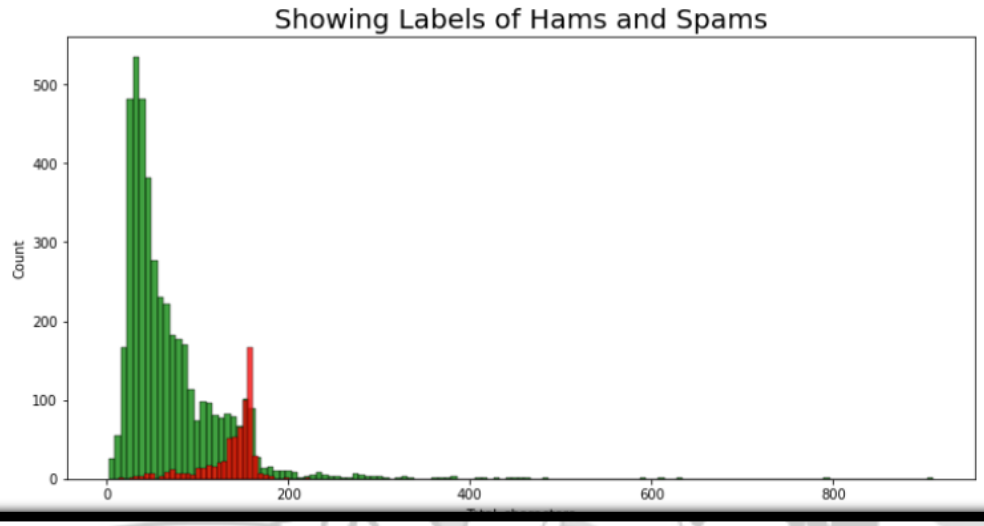
In [36]:
```
plt.figure(figsize=(28,5))
sns.countplot(x=df['Total_Sentences'],hue=df['Class_Labels'],data=df)
```

Out[36]:



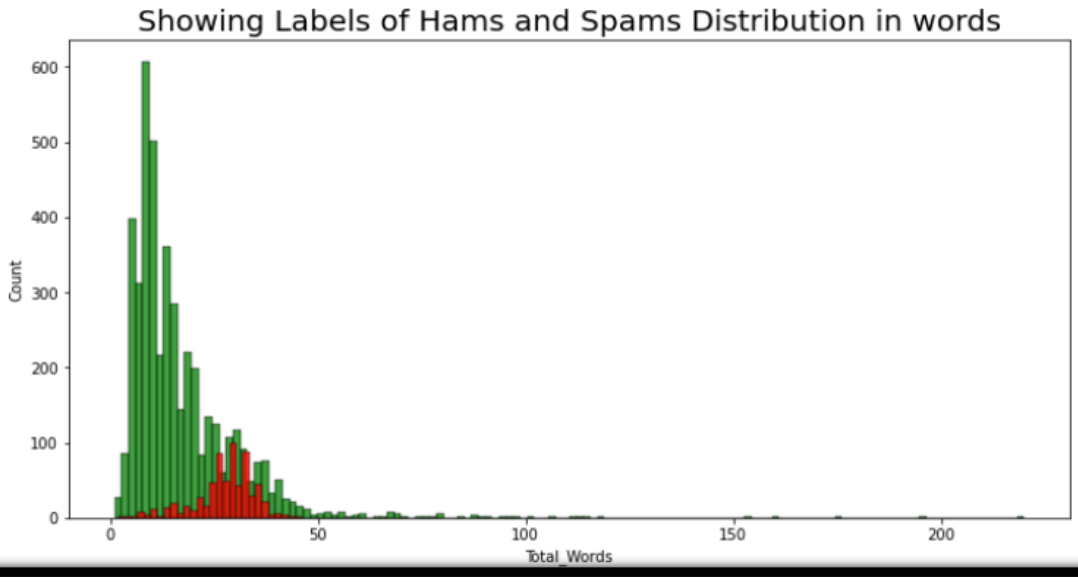
SPAM and HAM classification based on Total characters

In [40]:
```
plt.figure(figsize=(12,6))
sns.histplot(df[df['Class_Labels']==0]['Total_characters'],color='green')
sns.histplot(df[df['Class_Labels']==1]['Total_characters'],color='red')
plt.title("Showing Labels of Hams and Spams",fontsize=20)
plt.show()
```



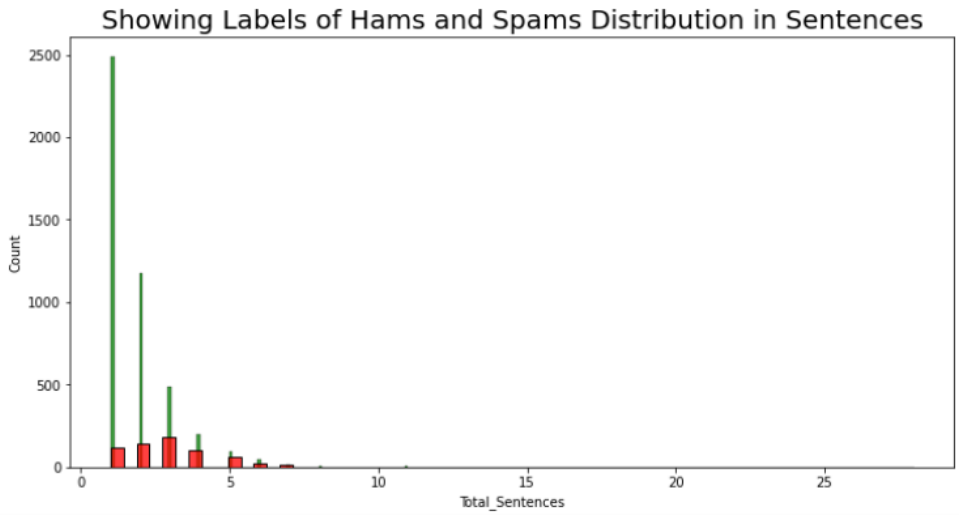
SPAM AND HAM CLASSIFICATION BASED ON WORDS

```
plt.figure(figsize=(12,6))
sns.histplot(df[df['Class_Labels']==0]['Total_Words'],color='green')
sns.histplot(df[df['Class_Labels']==1]['Total_Words'],color='red')
plt.title("Showing Labels of Hams and Spams Distribution in words",fontsize=20)
plt.show()
```

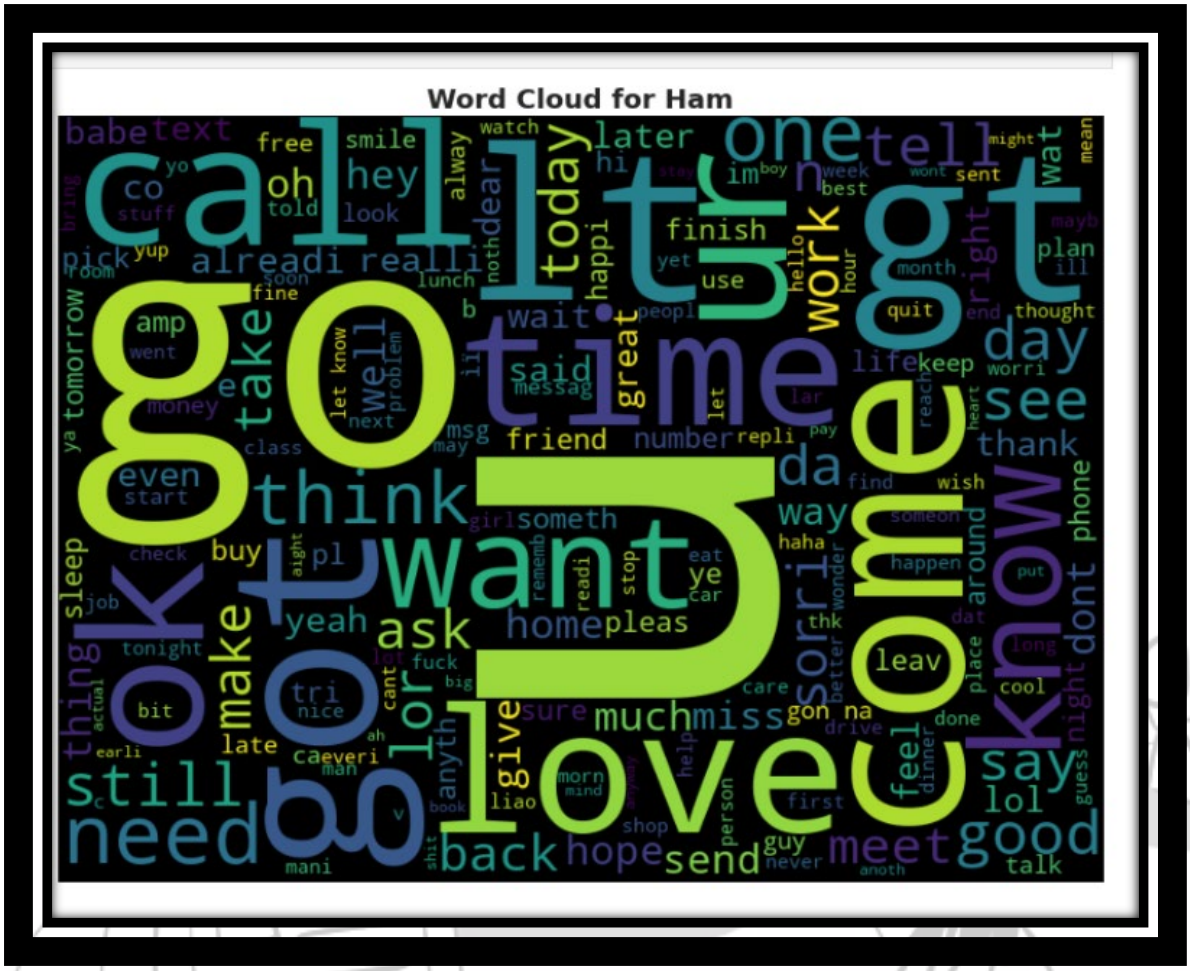


SPAM AND HAM CLASSIFICATION BASED ON SENTENCES

In [42]:
```
plt.figure(figsize=(12,6))
sns.histplot(df[df['Class_Labels']==0]['Total_Sentences'],color='green')
sns.histplot(df[df['Class_Labels']==1]['Total_Sentences'],color='red')
plt.title("Showing Labels of Hams and Spams Distribution in Sentences",fontsize=20)
plt.show()
```



Checking Frequent used texts in spam and ham messages

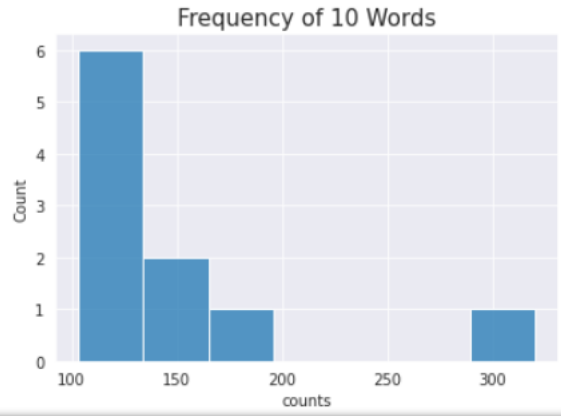


SPAM TOP 10 WORDS

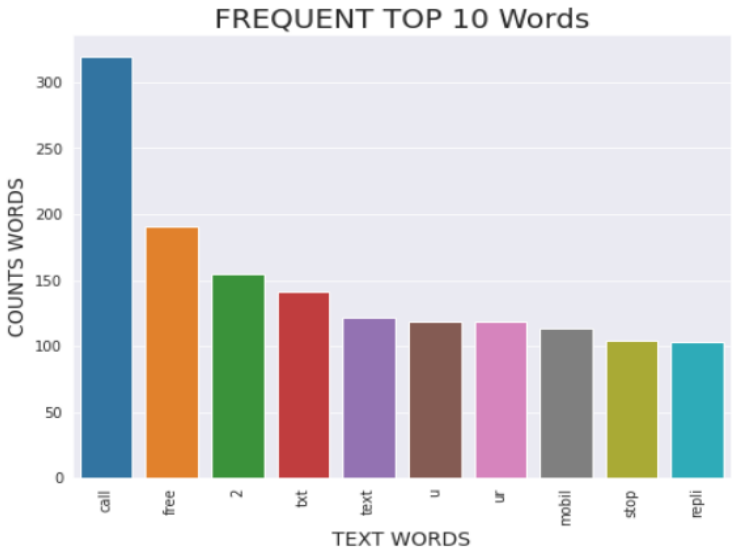
Out[93]:

	text	counts
0	call	320
1	free	191
2	2	155
3	txt	141
4	text	122
5	u	119
6	ur	119
7	mobil	114
8	stop	104
9	repli	103

```
sns.histplot(data=Top_Words,x=Top_Words['counts'])
plt.title("Frequency of 10 Words",fontsize=15)
plt.show()
```

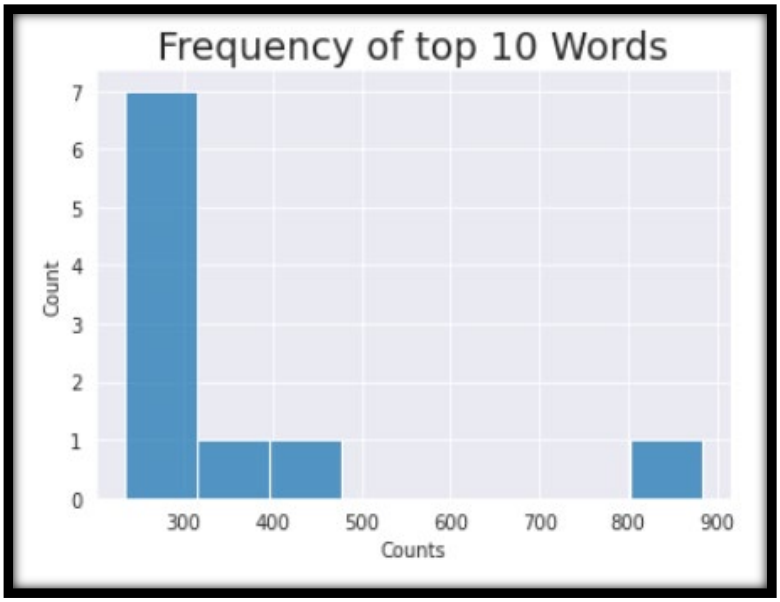


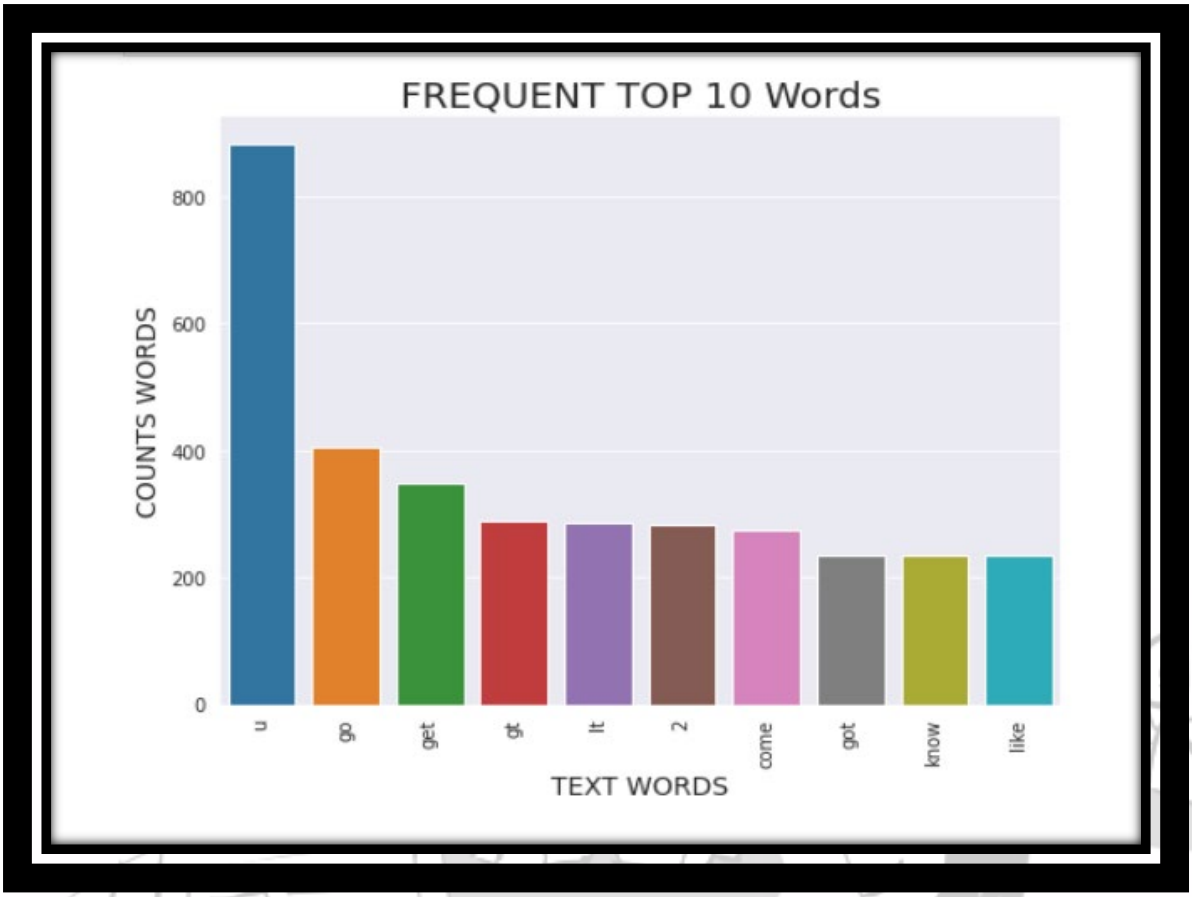
```
In [112…
plt.figure(figsize=(8,6))
sns.barplot(pd.DataFrame(Counter(spam_corpus).most_common(10))[0],pd.DataFrame(Counter(spa
plt.xlabel("TEXT WORDS",fontsize=14)
plt.xticks(rotation=90)
plt.ylabel("COUNTS WORDS",fontsize=14)
plt.title("FREQUENT TOP 10 Words",fontsize=20)
plt.show()
```



HAM CORPUS

	Text	Counts
0	u	883
1	go	404
2	get	349
3	gt	288
4	lt	287
5	2	284
6	come	275
7	got	236
8	know	236
9	like	234





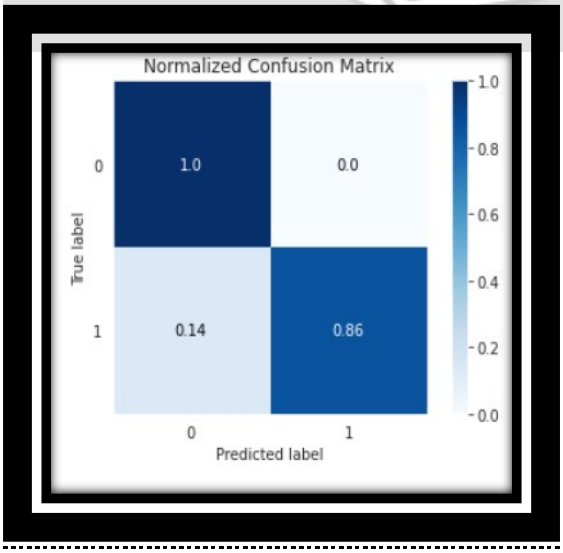
➢ Interpretation of the Results

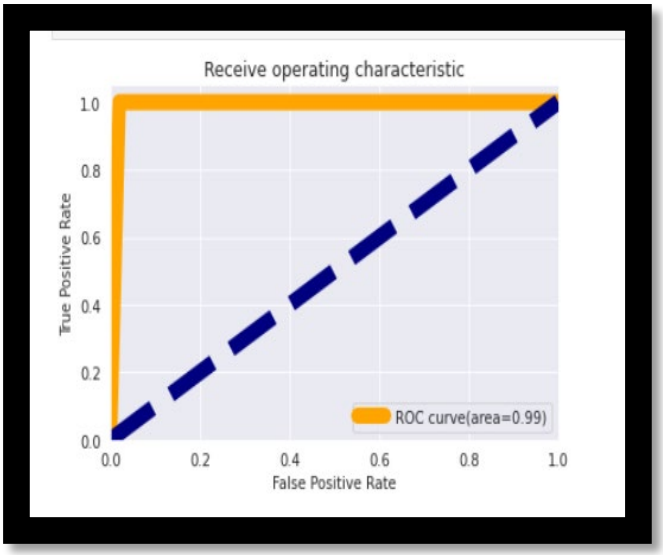
- This dataset was very special as it had a separate data types and result was drawn from object based independent variable.
- Firstly, the datasets were having Maximum null values. Hence no chance of imputation. Those were dropped.
- The data set had 403 duplicated rows. Checked and dropped.
- I found maximum object-based columns and no integer columns.
- I used maximum Count and Cat Plot followed by hist plot, Pearson correlation plot to find the relationship with target variable.
- I notice a huge number of outliers and high skewness in the data.
- We used lower function, string. Punction function, stemming for text pre-processing.
- We used Bag of Words for Count Vectorization.
- We used Word Clouds to detect the frequent used words in spam and HAM messages.

- We have to use multiple models Mainly based on Naïve Based Classifier, NLP classifiers and other Classifiers.
- Logistic Regression and MLP has best precision score, and accuracy score above 98 percent
- Finally selected Linear Regression Model and saved the model and finally printed the score and finally compared the data with the test data with the original data.

Hyper Parameter- Logistic Regression

```
Training Score 0.97726723095526 And Precision Score 100.0
Accuracy Training Score = 0.9961305925030229  Accuracy Test Score = 0.9825918762088974

Training Confusion_Matrix
 [[3609    1]
 [  15  510]] Testing Confusion_Matrix
 [[906   0]
 [ 18 110]]
Classification Report
              precision    recall  f1-score   support

           0       0.98      1.00      0.99       906
           1       1.00      0.86      0.92       128

    accuracy                           0.98      1034
   macro avg       0.99      0.93      0.96      1034
weighted avg       0.98      0.98      0.98      1034
```

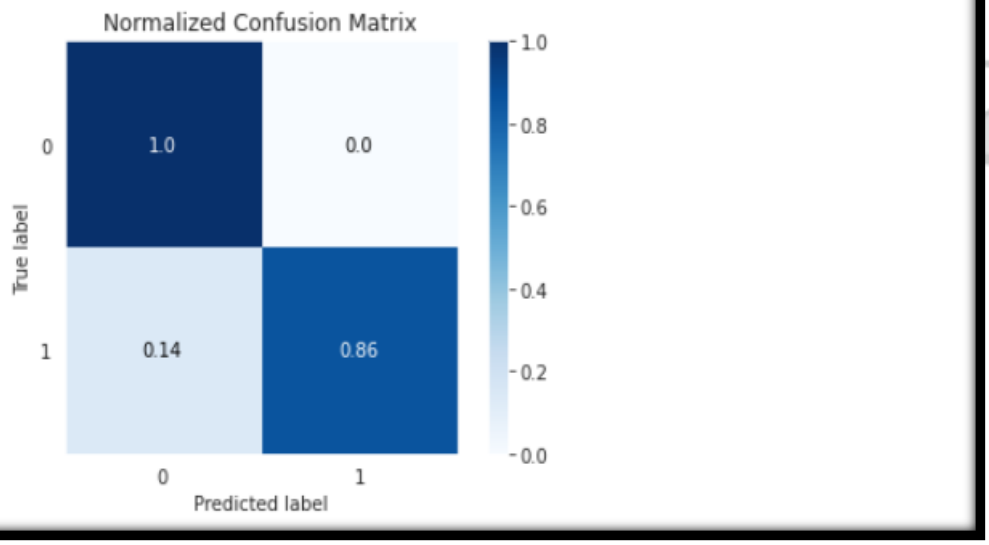


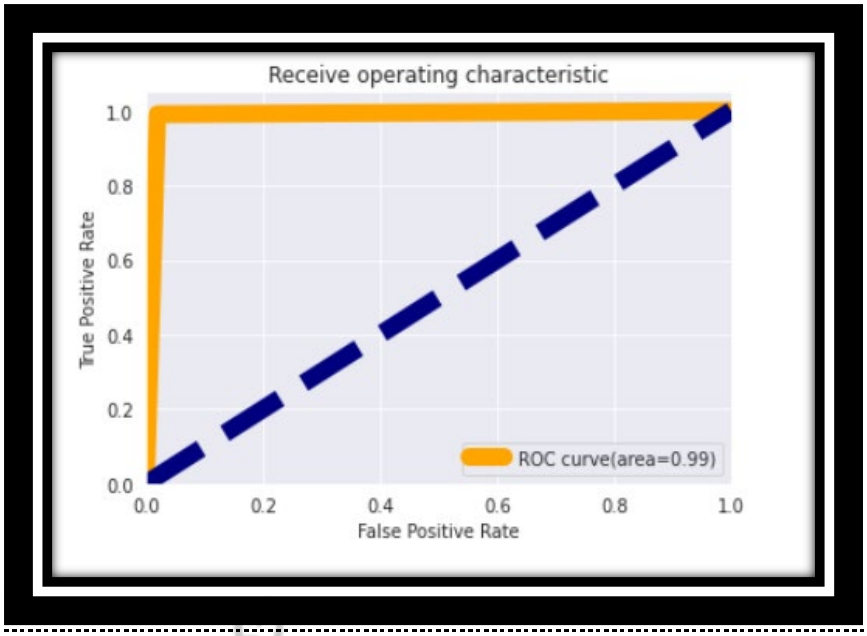


MLP CLASSIFIER:

```
Training Score 0.97726723095526 And Precision Score 99.09909909909909
Accuracy Training Score = 1.0  Accuracy Test Score = 0.9816247582205029

Training Confusion_Matrix
 [[3610    0]
 [   0  525]] Testing Confusion_Matrix
 [[905   1]
 [ 18 110]]
Classification Report
              precision    recall  f1-score   support

           0       0.98      1.00      0.99       906
           1       0.99      0.86      0.92       128

    accuracy                           0.98      1034
   macro avg       0.99      0.93      0.96      1034
weighted avg       0.98      0.98      0.98      1034
```

```
# Creating a normalized confusion matrix
import scikitplot as skplt
skplt.metrics.plot_confusion_matrix(y_test, pred_test1, normalize=True)
plt.show()
```





There was an increment of 0.20 with logistic model with 100 percent precision. We will save the the LOGISTIC MODEL:

SAVING MODEL

```
In [217…
import pickle
filename='spam_classification.pkl'
pickle.dump(lg,open(filename,"wb"))
```

Loading Model

```
In [218…
# loading pack file
pickled_model= pickle.load(open(filename,'rb'))
result=pickled_model.score(x_test,y_test)
result*100

Out[218…
98.25918762088975
```

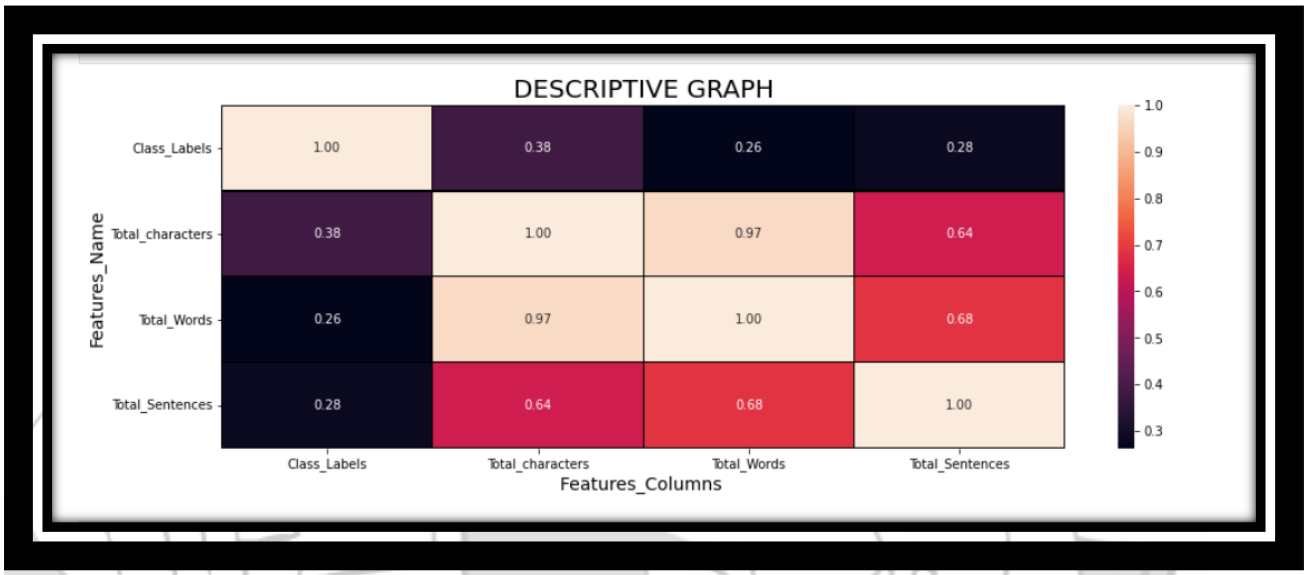
VERIFICATION WITH TESTING MODEL

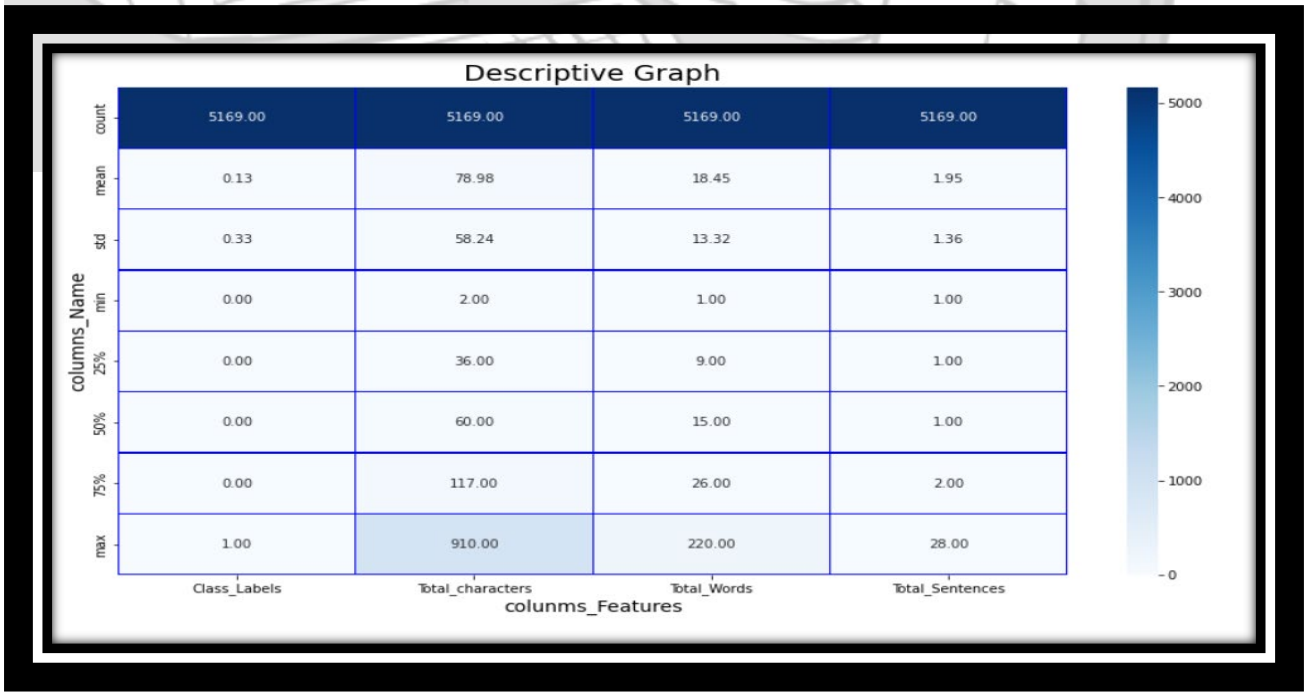
```
In [221…
array=np.array(y_test)
conclude=pd.DataFrame([pickled_model.predict(x_test)[:],y_test[:]],index=['Predicted','Original'])
conclude
```

Out[221…

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41
Predicted	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0
Original	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0

➢ OTHERS **1. Correlation Graph: 2. Descriptive Graph**





CONCLUSION

➢ Key Findings and Conclusions of the Study

By, the help of Messages in Data set I have made a SPAM CLASSIFICATION Model. We have done EDA, cleaned data and Visualized Data on the test Data Set. While cleaning the data Set, we analysed that 03 column is having Maximum Null Values So I dropped that column. After that we have done prediction on basis of Data using Data Pre-processing, Checked Correlation, removed Punctation stop words, extra space, leaning and trailing white space, converted text into vectors using count Vectorizer. Finally performed Train and Test on the extracted variable in that Data Set. We worked with various classifiers while using these given models and finally selected best model which was giving best accuracy, and Precession Score followed by F1 and CV score. Logistic Regression and MLP model were selected and sent for Hyper Tunning was used with Grid Search CV, And Best was Saved on the basis of Accuracy score and Precision Score at last I used that model to compare with predicted and Actual test data.

Thus, our project Stands completed **as filename 'spam_classification.pkl'**

➢ Learning Outcomes of the Study in respect of Data Science

I found that the dataset was quite interesting to handle as it contains all types of data in it. Improvement in computing technology has made it possible to examine social information that cannot previously be captured, processed and analysed. New analytical techniques of machine learning can be used in SMS Spam Classification.

The power of visualization has helped us in understanding the data by graphical representation it has made me to understand what data is trying to say. Data cleaning is one of the most important steps to remove missing value and to replace null value and zero values with their respective mean, median or mode.

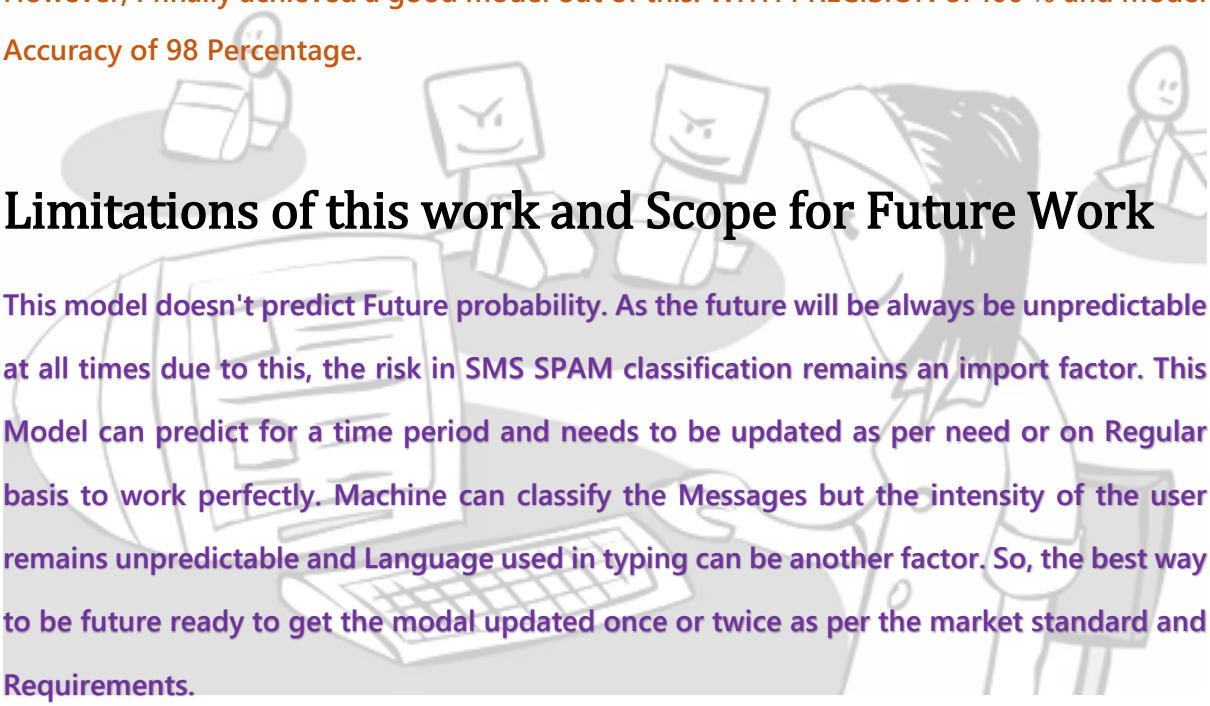
This study is an EXPLORATORY DATA ANALYSIS has attempted to use machine learning algorithms in finding the probability of Spam and Ham Messages in each model. To

conclude the application of Machine Learning in prediction is still at an early stage. I hope this study has moved a small step ahead in providing solution to the companies.

The Challenges I faced was when I was not much aware of NLP and I took the help for Md Kashif Sir and used various sources like YOU-Tube, Kaggle and Medium and NLTK library where I faced problem. Even The Algorithm was working huge time to complete the test and CV_Score generating was again time taking in each of the cases.

Finally, I had to run the test twice and thrice with different standardization process. I was actually thinking to get best out of those.

However, I finally achieved a good model out of this. WITH PRECISION of 100 % and Model Accuracy of 98 Percentage.



Limitations of this work and Scope for Future Work

This model doesn't predict Future probability. As the future will be always be unpredictable at all times due to this, the risk in SMS SPAM classification remains an import factor. This Model can predict for a time period and needs to be updated as per need or on Regular basis to work perfectly. Machine can classify the Messages but the intensity of the user remains unpredictable and Language used in typing can be another factor. So, the best way to be future ready to get the modal updated once or twice as per the market standard and Requirements.

Thank-you