
# Email Spam Classifier

Submitted by:

Aniruddha Sawant

## ACKNOWLEDGMENT

Aniruddha Sawant (“I”) acknowledge that I have used the data and files provided by Flip Robo (“Company”) namely ‘Problem Statement’ and ‘spam.csv’ to build the predictive model and have used ‘sample documentation’ for guidance and to write report.

# INTRODUCTION

- **Background of the Domain Problem: -**

  Spam Detector is used to detect unwanted, malicious and virus infected texts and helps to separate them from the nonspam texts. It uses a binary type of classification containing the labels such as 'ham' (nonspam) and spam. Application of this can be seen in Google Mail (GMAIL) where it segregates the spam emails in order to prevent them from getting into the user's inbox.

- **Review of Literature: -**

  The files contain one message per line. Each line is composed by two columns: v1 contains the label (ham or spam) and v2 contains the raw text. A collection of 5573 rows SMS spam messages was manually extracted from the Grumbletext 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. The identification of the text of spam messages in the claims is a very hard and time-consuming task, and it involved carefully scanning hundreds of web pages. 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. The messages largely originate from Singaporeans and mostly from students attending the University. These messages were collected from volunteers who were made aware that their contributions were going to be made publicly available.

- **Motivation for the Problem Undertaken: -**

  We are required to classify the email whether it's a spam or ham with the available mails and messages. This model will then be used by clients to understand how exactly the emails vary with the variables. Further, the model will be a good way for the management to understand the mail dynamics.

# Analytical Problem Framing

- **Mathematical/ Analytical Modelling of the Problem**

1. Data structuring
2. Analysing the data
3. Cleaning and processing the data
4. Writing down findings and observations
5. Using different models to train the data
6. Selecting best fitted model for predictions
7. Hyper tuning the selected model
8. Predicting outcome for test data

- **Data Sources and their formats**

We have an excel sheet that contains the messages and mails with the label that contains ham and spam. Below is the sample of the data.

| v1 | v2 |
|---|---|
| ham | Go until jurong point, crazy.. Available only in bugis n great world la e buffet... Cine there got amore wat... |
| ham | Ok lar... Joking wif u oni... |
| spam | Free entry in 2 a wkly comp to win FA Cup final tkts 21st May 2005. Text FA to 87121 to receive entry question(std txt rate)T&C's apply 08452810075over18's |
| ham | U dun say so early hor... U c already then say... |
| ham | Nah I don't think he goes to usf, he lives around here though |
| spam | FreeMsg Hey there darling it's been 3 week's now and no word back! I'd like some fun you up for it still? Tb ok! XxX std chgs to send, å£1.50 to rcv |
| ham | Even my brother is not like to speak with me. They treat me like aids patent. |
| ham | As per your request 'Melle Melle (Oru Minnaminunginte Nurungu Vettam)' has been set as your callertune for all Callers. Press *9 to copy your friends Callertune |
| spam | WINNER!! As a valued network customer you have been selected to receivea å£900 prize reward! To claim call 09061701461. Claim code KL341. Valid 12 hours only. |
| spam | Had your mobile 11 months or more? U R entitled to Update to the latest colour mobiles with camera for Free! Call The Mobile Update Co FREE on 08002986030 |
| ham | I'm gonna be home soon and i don't want to talk about this stuff anymore tonight, k? I've cried enough today. |

```
1 df
```

| | v1 | v2 | Unnamed: 2 | Unnamed: 3 | Unnamed: 4 |
|---|---|---|---|---|---|
| **0** | ham | Go until jurong point, crazy.. Available only ... | NaN | NaN | NaN |
| **1** | ham | Ok lar... Joking wif u oni... | NaN | NaN | NaN |
| **2** | spam | Free entry in 2 a wkly comp to win FA Cup fina... | NaN | NaN | NaN |
| **3** | ham | U dun say so early hor... U c already then say... | NaN | NaN | NaN |
| **4** | ham | Nah I don't think he goes to usf, he lives aro... | NaN | NaN | NaN |
| **...** | ... | ... | ... | ... | ... |
| **5567** | spam | This is the 2nd time we have tried 2 contact u... | NaN | NaN | NaN |
| **5568** | ham | Will Ì_ b going to esplanade fr home? | NaN | NaN | NaN |
| **5569** | ham | Pity, * was in mood for that. So...any other s... | NaN | NaN | NaN |
| **5570** | ham | The guy did some bitching but I acted like i'd... | NaN | NaN | NaN |
| **5571** | ham | Rofl. Its true to its name | NaN | NaN | NaN |

- **Data Pre-processing**

1. Clean and organized the data.
2. Checked the data type of each column.
3. Changed the Object data type to Integer.
4. Checked whether the data has any Null Values.
5. Checked whether the data is categorical data or continuous data.
6. Encoded remaining Object data type columns to integer using encoder technique.
7. Checked the co-relation of features with label i.e v1.
8. Checked the Distribution of data.

- **Data Inputs**

1. V1 – This is a label that contains whether the message or mail is ham or a spam
2. V2 – This column contains all the messages and mails.

- **Hardware and Software Requirements and Tools Used**
  1. Libraries and packages used
     - import numpy as np – For Numpy work
     - import pandas as pd – To work on DataFrame
     - import seaborn as sns – Plotting Graphs
     - import matplotlib.pyplot as plt - Plotting Graphs
     - import pickle – To save the Model
     - from sklearn.preprocessing import LabelEncoder (To encode object data to Integer)
     - enc = LabelEncoder()= Assigned LabelEncoder to variable
     - from sklearn.model_selection import train_test_split – To split the data into train and test, GridSearchCV = to search the best fit parameters
     - from sklearn.metrics import accuracy_score, classification_report, confusion_matrix, precision_score, roc_curve, roc_auc_score, f1_score – To calculate and analyse model metrics.
     - import warnings, warnings.filterwarnings('ignore') – To ignore unwanted Warnings
  2. Machine Learning models used
     - from sklearn.ensemble import RandomForestClassifier
     - from sklearn.linear_model import LogisticRegression
     - from sklearn.ensemble import GradientBoostingClassifier
     - from sklearn.tree import DecisionTreeClassifier
     - from sklearn.neighbors import KNeighborsClassifier
     - from sklearn.svm import SVC
     - from sklearn.naive_bayes import MultinomialNB, GaussianNB, BernoulliNB
  3. NLP Libraries and packages used
     - import nltk
     - import string
     - from nltk.corpus import stopwords
     - from nltk.tokenize import word_tokenize
     - from nltk.stem import PorterStemmer
     - from wordcloud import WordCloud
     - from sklearn.feature_extraction.text import CountVectorizer
     - import re
  4. Hardware used – 11th Gen Intel(R) Core (TM) i3-1115G4 @ 3.00GHz 3.00 GHz with 8.00 GB RAM and Windows 11
  5. Software used – Anaconda and Jupyter Notebook to build the model and Excel to clean and structured the data.

# Model/s Development and Evaluation

- ## Identification of possible problem

1. The data was not structured and organized hence cleaned the data using various data cleaning and pre-processing techniques.
2. Have run several EDAs to understand the data.
3. The data is not balanced since spam comments are less that the ham comments hence have changed the evaluation methods, instead of focusing on accuracy I focused on precision score.
4. Used vectorization to build the model.

- ## Testing of Identified Approaches

These are the algorithms which have been used to train and test data.

1. RandomForestClassifier
2. LogisticRegression
3. GradientBoostingClassifier
4. DecisionTreeClassifier
5. KNeighborsClassifier
6. Support Vector Classifier
7. GaussianNB
8. MultinomialNB
9. BernoulliNB

- **Run and evaluate selected models**

1. **GaussianNB**: -

```
gnb = GaussianNB()
gnb.fit(x_train,y_train)

print_score(gnb,x_train,x_test,y_train,y_test, train=True)
print_score(gnb,x_train,x_test,y_train,y_test, train=False)
```

```
===============Train Result===============
Accuracy Score: 92.70%

===============Test Result===============
Accuracy Score: 88.63%


 Test Classification Report
              precision    recall  f1-score   support

           0       0.98      0.89      0.93      1134
           1       0.52      0.89      0.66       159

    accuracy                           0.89      1293
   macro avg       0.75      0.89      0.80      1293
weighted avg       0.93      0.89      0.90      1293


_______________________________________

confusion Matrix-
 [[1004  130]
 [  17  142]]
```

2. **MultinomialNB**: -

```
mnb = MultinomialNB()
mnb.fit(x_train,y_train)

print_score(mnb,x_train,x_test,y_train,y_test, train=True)
print_score(mnb,x_train,x_test,y_train,y_test, train=False)
```

```
===============Train Result===============
Accuracy Score: 99.05%

===============Test Result===============
Accuracy Score: 98.14%


 Test Classification Report
              precision    recall  f1-score   support

           0       0.99      0.98      0.99      1133
           1       0.90      0.96      0.93       160

    accuracy                           0.98      1293
   macro avg       0.94      0.97      0.96      1293
weighted avg       0.98      0.98      0.98      1293


_______________________________________

confusion Matrix-
 [[1115   18]
 [   6  154]]
```

## 3. **BernoulliNB**: -

```
bnb = BernoulliNB()
bnb.fit(x_train,y_train)

print_score(bnb,x_train,x_test,y_train,y_test, train=True)
print_score(bnb,x_train,x_test,y_train,y_test, train=False)
```

```
===============Train Result===============
Accuracy Score: 97.55%

===============Test Result===============
Accuracy Score: 98.07%


 Test Classification Report
              precision    recall  f1-score   support

           0       0.98      0.99      0.99      1152
           1       0.95      0.87      0.91       141

    accuracy                           0.98      1293
   macro avg       0.97      0.93      0.95      1293
weighted avg       0.98      0.98      0.98      1293


========================================

confusion Matrix-
 [[1146    6]
 [  19  122]]
```

## 4. **LogisticRegression**: -

```
log = LogisticRegression()
log.fit(x_train,y_train)

print_score(log,x_train,x_test,y_train,y_test, train=True)
print_score(log,x_train,x_test,y_train,y_test, train=False)
```

```
===============Train Result===============
Accuracy Score: 99.43%

===============Test Result===============
Accuracy Score: 98.69%


 Test Classification Report
              precision    recall  f1-score   support

           0       0.99      1.00      0.99      1152
           1       0.98      0.90      0.94       141

    accuracy                           0.99      1293
   macro avg       0.98      0.95      0.96      1293
weighted avg       0.99      0.99      0.99      1293


========================================

confusion Matrix-
 [[1149    3]
 [  14  127]]
```

5. **KNeighborsClassifier**: -

```
knn = KNeighborsClassifier()
knn.fit(x_train,y_train)

print_score(knn,x_train,x_test,y_train,y_test, train=True)
print_score(knn,x_train,x_test,y_train,y_test, train=False)
```

```
===============Train Result===============
Accuracy Score: 92.03%

===============Test Result===============
Accuracy Score: 92.50%


 Test Classification Report
              precision    recall  f1-score   support

           0       0.92      1.00      0.96      1152
           1       1.00      0.31      0.48       141

    accuracy                           0.92      1293
   macro avg       0.96      0.66      0.72      1293
weighted avg       0.93      0.92      0.91      1293


========================================

confusion Matrix-
 [[1152    0]
 [  97   44]]
```

6. **GradientBoostingClassifier**: -

```
gbdt = GradientBoostingClassifier()
gbdt.fit(x_train,y_train)

print_score(gbdt,x_train,x_test,y_train,y_test, train=True)
print_score(gbdt,x_train,x_test,y_train,y_test, train=False)
```

```
===============Train Result===============
Accuracy Score: 97.57%

===============Test Result===============
Accuracy Score: 96.44%


 Test Classification Report
              precision    recall  f1-score   support

           0       0.97      0.99      0.98      1152
           1       0.92      0.74      0.82       141

    accuracy                           0.96      1293
   macro avg       0.94      0.86      0.90      1293
weighted avg       0.96      0.96      0.96      1293


========================================

confusion Matrix-
 [[1143    9]
 [  37  104]]
```

## 7. SupportVectorClassifier: -

```
svc = SVC()
svc.fit(x_train,y_train)

print_score(svc,x_train,x_test,y_train,y_test, train=True)
print_score(svc,x_train,x_test,y_train,y_test, train=False)
```

```
===============Train Result===============
Accuracy Score: 99.46%

===============Test Result===============
Accuracy Score: 97.68%


 Test Classification Report
              precision    recall  f1-score   support

           0       0.97      1.00      0.99      1152
           1       1.00      0.79      0.88       141

    accuracy                           0.98      1293
   macro avg       0.99      0.89      0.93      1293
weighted avg       0.98      0.98      0.98      1293


=========================================

confusion Matrix-
 [[1152    0]
 [  30  111]]
```

## 8. DecisionTreeClassifier: -

```
dtc = DecisionTreeClassifier()
dtc.fit(x_train,y_train)

print_score(dtc,x_train,x_test,y_train,y_test, train=True)
print_score(dtc,x_train,x_test,y_train,y_test, train=False)
```

```
===============Train Result===============
Accuracy Score: 100.00%

===============Test Result===============
Accuracy Score: 96.21%


 Test Classification Report
              precision    recall  f1-score   support

           0       0.97      0.98      0.98      1152
           1       0.86      0.78      0.82       141

    accuracy                           0.96      1293
   macro avg       0.92      0.88      0.90      1293
weighted avg       0.96      0.96      0.96      1293


=========================================

confusion Matrix-
 [[1134   18]
 [  31  110]]
```

## 9. RandomForestClassifier: -

```
rfc = RandomForestClassifier()
rfc.fit(x_train,y_train)

print_score(rfc,x_train,x_test,y_train,y_test, train=True)
print_score(rfc,x_train,x_test,y_train,y_test, train=False)
```

```
===============Train Result===============
Accuracy Score: 100.00%

===============Test Result===============
Accuracy Score: 97.68%


 Test Classification Report
              precision    recall  f1-score   support

           0       0.98      1.00      0.99      1152
           1       0.98      0.80      0.88       141

    accuracy                           0.98      1293
   macro avg       0.98      0.90      0.93      1293
weighted avg       0.98      0.98      0.98      1293


========================================

confusion Matrix-
 [[1150    2]
 [  28  113]]
```

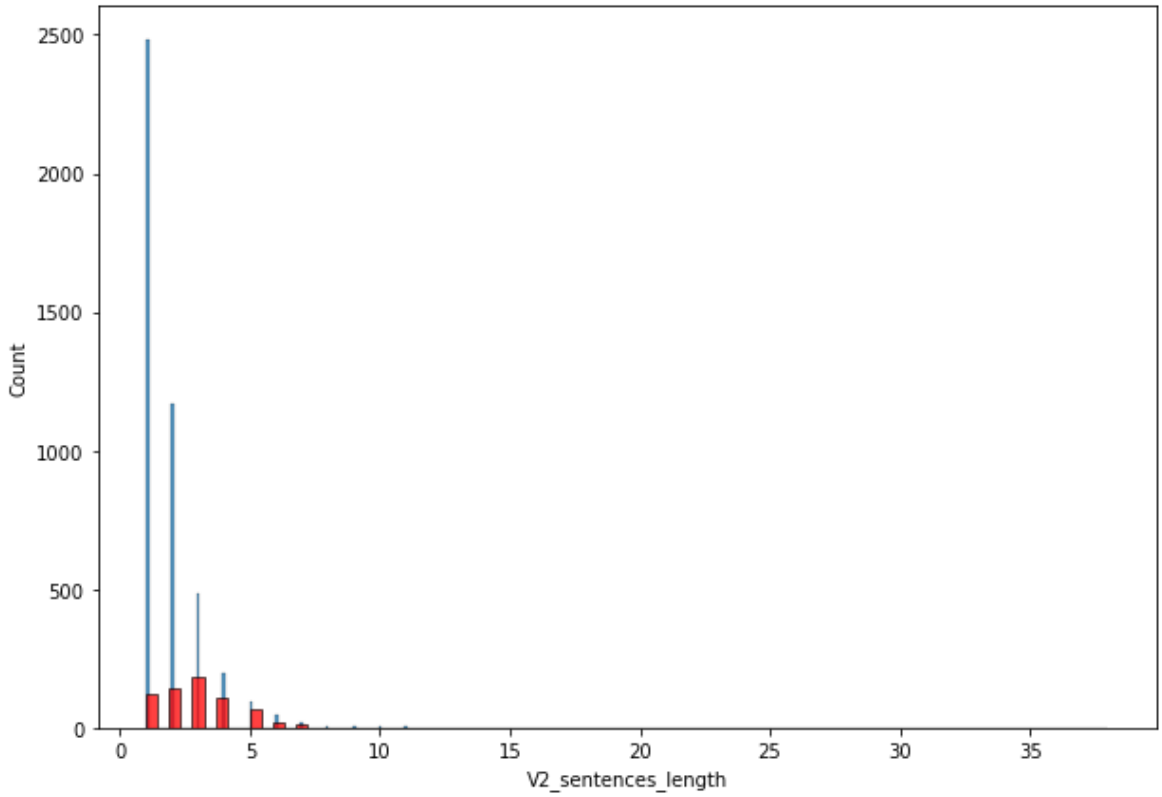
- **Visualizations**
  1. **WordCloud: -**


**Observations** -

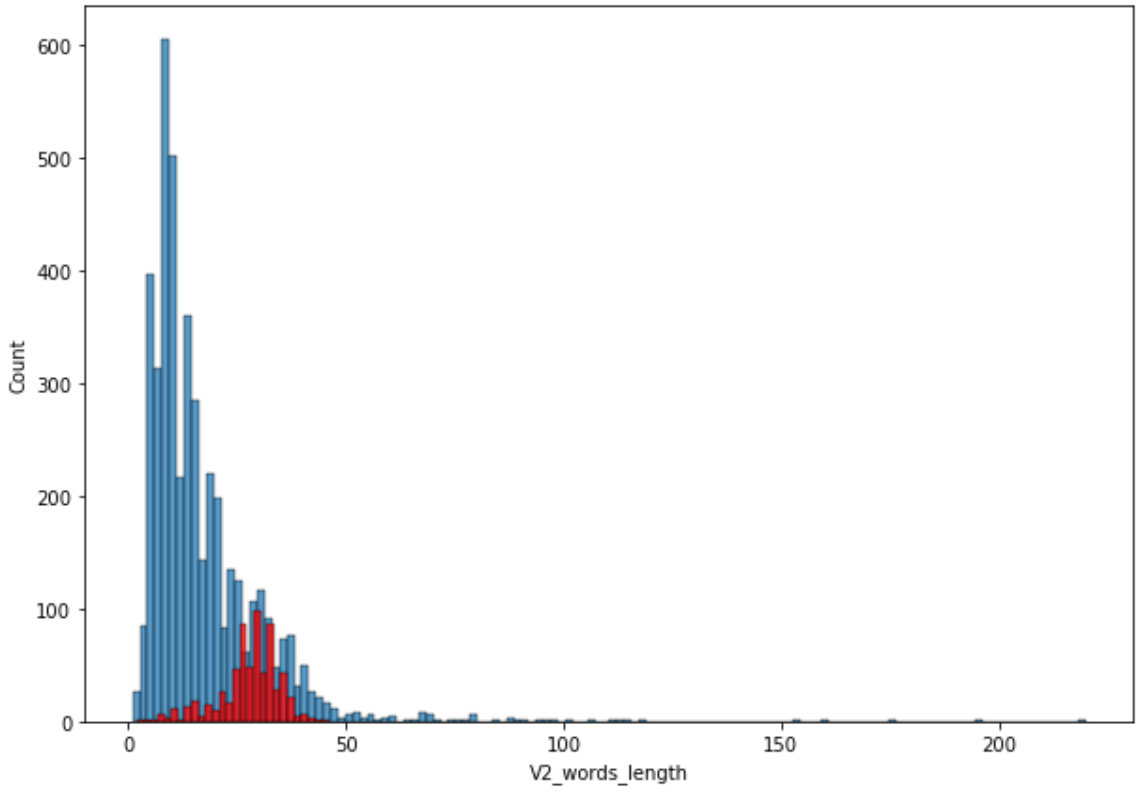
Free, Call, Please, Text, txt are some words which are occurring for most number of time

## 2. Histogram: -

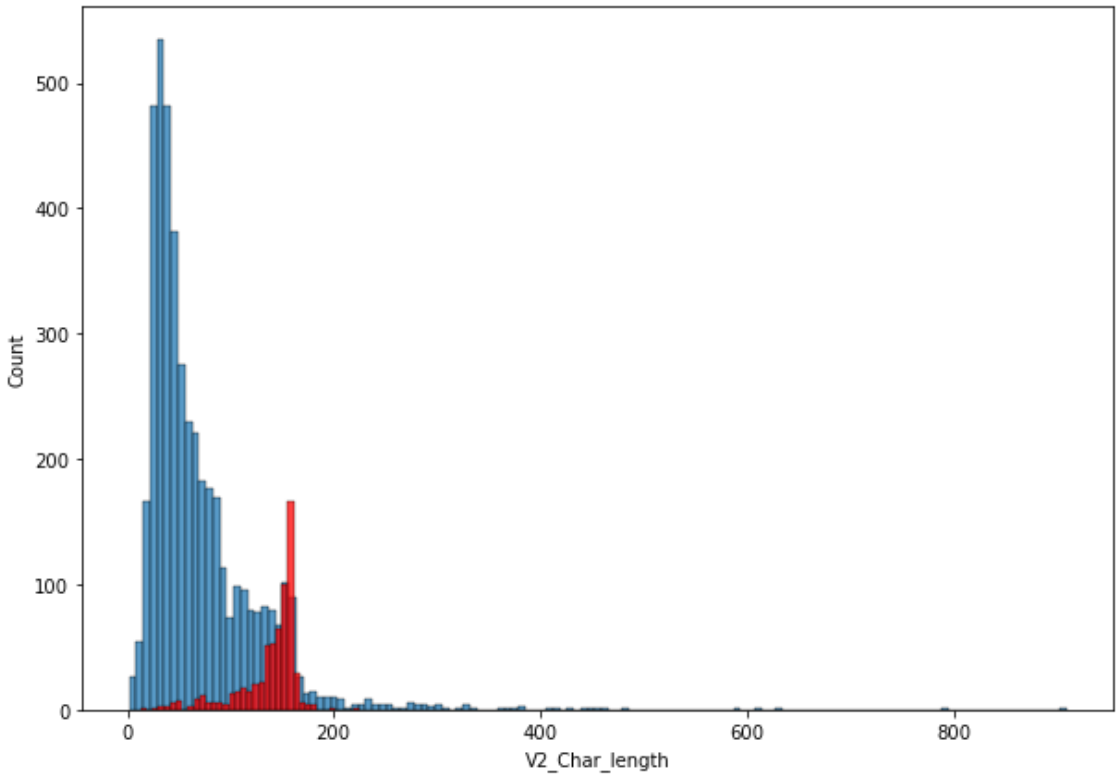
1. **V2_sentences_length –** Showing the distribution of sentence length of each row.




2. **V2_words_length –** Showing the distribution of word length of each row.

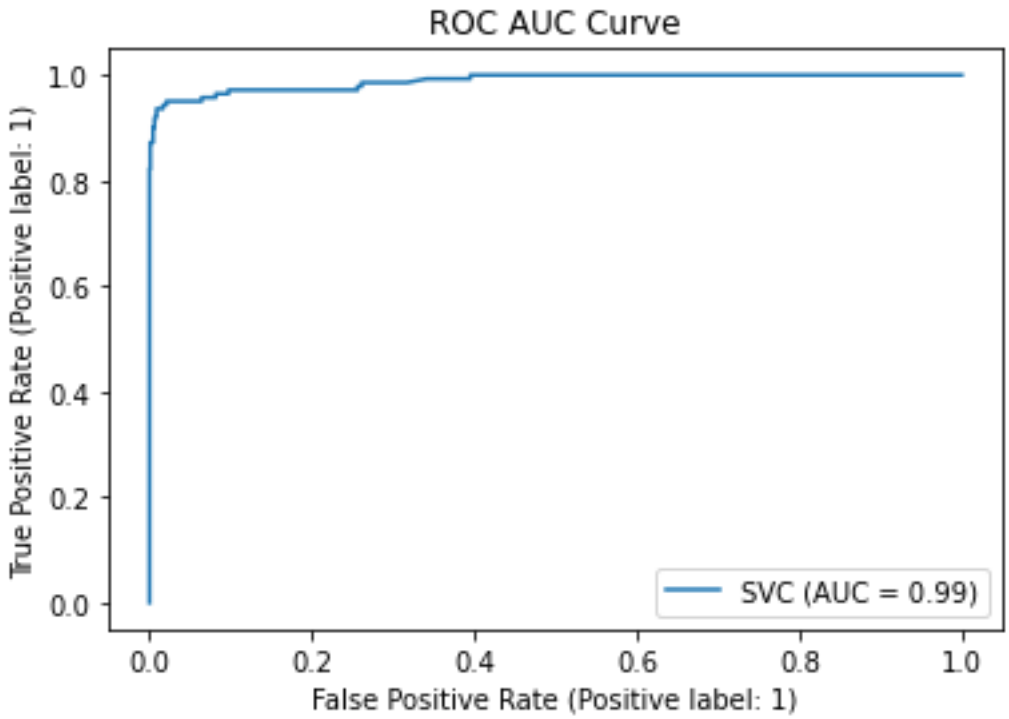


3. **V2_Char_length –** Showing the distribution of character length of each row.




- **Observation**
  1. Spam mails and messages contain a greater number of sentences that ham mails and messages.
  2. Spam mails and messages contain a greater number of words that ham mails and messages.
  3. Spam mails and messages contain a greater number of characters that ham mails and messages.

### 3. ROC AUC Curve: -




**Observations**:

- AUC Score is 99% which is pretty good

- **Interpretation of the Results**

Below is the table that shows the final result of the various models.

| Model_Name | Accuracy | Precision_Score | Confusion_Matrix_FP |
|---|---|---|---|
| SupportVectorClassifier | 97.68 | 1 | 0 |
| KNeighborsClassifier | 92.5 | 1 | 0 |
| RandomForestClassifier | 97.68 | 0.98 | 2 |
| LogisticRegression | 98.69 | 0.98 | 3 |
| BernoulliNB | 98.07 | 0.95 | 6 |
| GradientBoostingClassifier | 96.44 | 0.92 | 9 |
| MultinomialNB | 98.14 | 0.9 | 18 |
| DecisionTreeClassifier | 96.21 | 0.86 | 18 |
| GaussianNB | 88.63 | 0.52 | 130 |

# CONCLUSION

- ## Key Findings and Conclusions of the Study
  1. Selecting SupportVectorClassifier model since the Accuracy score i.e., 99.46% and test scores i.e., 97.68% are greater and close to each other.
  2. recall and f1-score are also high for SupportVectorClassifier model.
  3. Precision score is 1 for this model means there are no false positives in the test result which states that our model is working really well.

- ## Learning Outcomes of the Study in respect of Data Science
  1. Data Cleaning helps to convert unorganized and unstructured data into structured data which will be used to make findings.
  2. Data visualization helps understand and analyse the data.
  3. Model building helps to predict outcomes, in this case SupportVectorClassifier model fits perfect for this dataset.
  4. There are many unwanted characters in the messages and mails which need more pre-processing.

- ## Limitations of this work and Scope for Future Work
  1. There are limitations in case the messages and mails are in other languages.
  2. There are so many unwanted characters, emails, emojis and links present in the data which need more processing.
  3. It is necessary to keep an eye on new and updated data to further train the model and make decision as per new data.