Email Spam Classifier

Submitted by:

Shruti Raj

ACKNOWLEDGMENT

I would like to express my gratitude to my guide Shwetank Mishra (SME, Flip Robo) for his constant guidance, continuous encouragement and unconditional help towards the development of the project. It was he who helped me whenever I got stuck somewhere in between. The project would have not been completed without his support and confidence he showed towards me.

Lastly, I would l like to thank all those who helped me directly or indirectly toward the successful completion of the project.

INTRODUCTION

- ## Business Problem Framing

  Email system is one of the most effective and commonly used sources of communication. The reason of the popularity of email system lies in its cost effective and faster communication nature. Unfortunately, email system is getting threatened by spam emails. Spam emails are the uninvited emails sent by some unwanted users also known as spammers with the motive of making money.

  The email users spend most of their valuable time in sorting these spam mails.

  Multiple copies of same message are sent many times which not only affect an organization financially but also irritates the receiving user.

  Spam emails are not only intruding the user's emails but they are also producing large amount of unwanted data and thus affecting the network's capacity and usage. In this paper, a Spam Mail Detection (SMD) system is proposed which will classify email data into spam and ham emails. The process of spam filtering focuses on three main levels: the email address, subject and content of the message.

  All mails have a common structure i.e., subject of the email and the body of the email. A typical spam mail can be classified by filtering its content. The process of spam mail detection is based on the assumption that the content of the spam mail is different than the legitimate or ham mail. For example, words related to the advertisement of any product, endorsement of services, dating related content etc. The process of spam email detection can be broadly categorized into two approaches: knowledge engineering and machine learning approach.

  Knowledge engineering is a network-based approach in which IP (internet protocol) address, network address along with some sets of defined rules are considered for the email classification. The approach has shown promising results but it is very time consuming. The maintenance and task of updating rules is not convenient for all users. On the other hand, machine learning approach does not involve any set of rules and is efficient than knowledge engineering approach.

  The classification algorithm classifies the email based on the content and other attributes.

- ## Conceptual Background of the Domain Problem

  Online platforms and social media become the place where people share the thoughts freely without any partiality and overcoming all the race people share their thoughts and ideas among the crowd.

Social media is a computer-based technology that facilitates the sharing of ideas, thoughts, and information through the building of virtual networks and communities. By design, social media is Internet-based and gives users quick electronic communication of content. Content includes personal information, documents, videos, and photos. Users engage with social media via a computer, tablet, or smartphone via web-based software or applications.

While social media is ubiquitous in America and Europe, Asian countries like India lead the list of social media usage. More than 3.8 billion people use social media.

In this huge online platform or an online community there are some people or some motivated mob wilfully bully others to make them not to share their thought in rightful way. They bully others in a foul language which among the civilized society is seen as ignominy. And when innocent individuals are being bullied by these mob these individuals are going silent without speaking anything. So, ideally the motive of this disgraceful mob is achieved.

To solve this problem, we are now building a model that identifies all the foul language and foul words, using which the online platforms like social media principally stops these mob using the foul language in an online community or even block them or block them from using this foul language.

- ## Review of Literature

The purpose of the literature review is to:

1. Identify the foul words or foul statements that are being used.
2. Stop the people from using these foul languages in online public forum.

To solve this problem, we are now building a model using our machine language technique that identifies all the foul language and foul words, using which the online platforms like social media principally stops these mob using the foul language in an online community or even block them or block them from using this foul language.

I have used 5 different Classification algorithms and shortlisted the best on basis on the metrics of performance and I have chosen one algorithm and build a model in that algorithm.

- Motivation for the Problem Undertaken

I am doing this for practice, to get more hands-on data exploration, Feature extraction and Model building.

Analytical Problem Framing

- Mathematical/ Analytical Modelling of the Problem

I start analysis on this project in importing the data set and simple play around with the data and identifying the characteristics of each column.

```
df = pd.read_csv('spam.csv',encoding='ISO-8859-1')
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

5572 rows × 5 columns

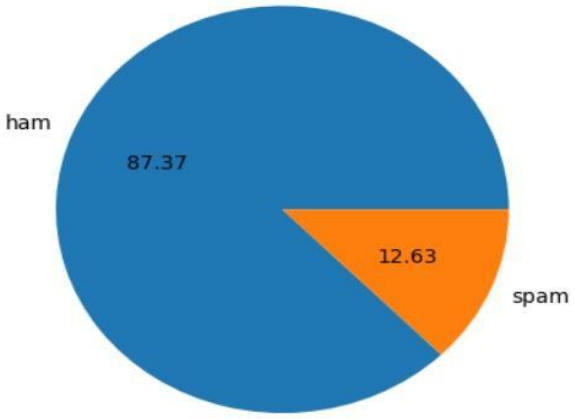
In the first stoke of analysis I understood that there are 5 columns in total in which 2 are numerical column and ‘Unnamed: 2’, ‘Unnamed: 3’, ‘Unnamed: 4’ data which has all unique values “connect text” have string values.

Since ‘Unnamed: 2’, ‘Unnamed: 3’ and ‘Unnamed: 4’ have all unique values, it won’t be helpful in analysis to I have dropped ‘Unnamed: 2’, ‘Unnamed: 3’, ‘Unnamed: 4’ column.

```
# drop unnecssary last 3 columns
df.drop(columns=['Unnamed: 2','Unnamed: 3','Unnamed: 4'],inplace=True)
```

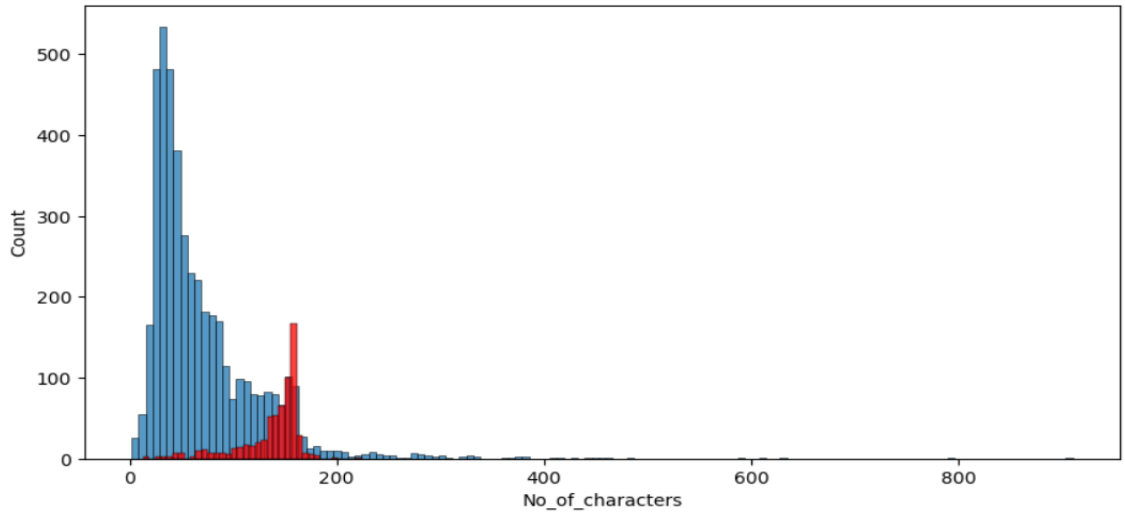
As we see below Ham sentence are 87.37% and Spam are 12.63%.

```
import matplotlib.pyplot as plt
plt.pie(df['Target'].value_counts(), labels=['ham','spam'],autopct="%0.2f")
plt.show()
df['Target'].value_counts()
```




```
0    4516
1     653
Name: Target, dtype: int64
```

```
plt.figure(figsize=(10,5))
sns.histplot(df[df['Target'] == 0]['No_of_characters'])
sns.histplot(df[df['Target'] == 1]['No_of_characters'],color='red')
plt.show()
```

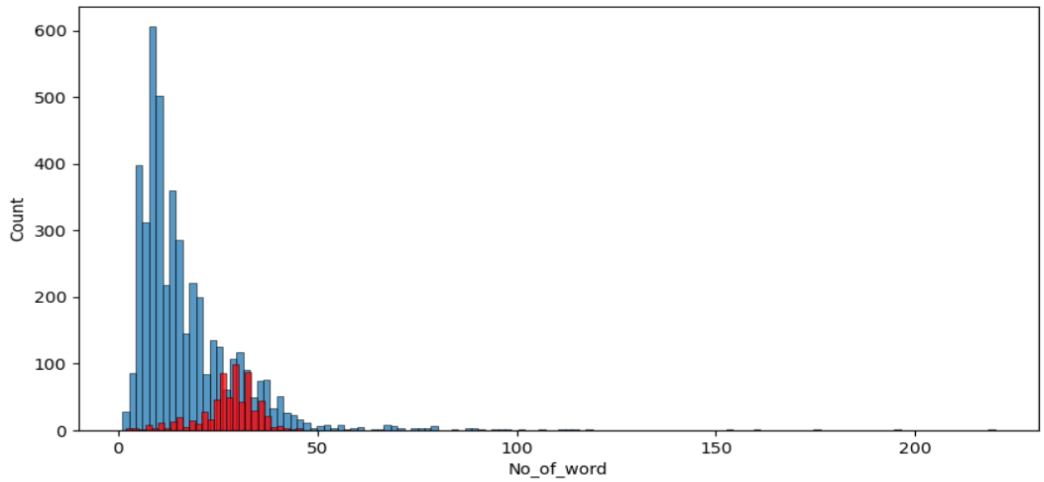


Comment:

- Number of characters in Spam SMS is comparatively much high than Ham SMS.

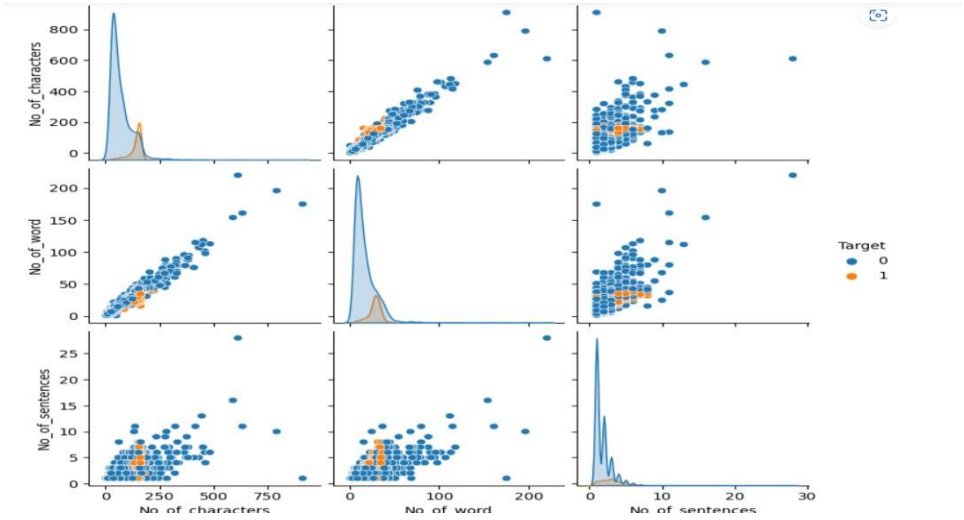
- On average each ham SMS contain 71 character, 17 words and 2 sentences.

```
plt.figure(figsize=(10,5))
sns.histplot(df[df['Target'] == 0]['No_of_word'])
sns.histplot(df[df['Target'] == 1]['No_of_word'],color='red');
```



Comment:

- Number of Word in Spam SMS is comparatively much high than Ham SMS.
- On average each spam SMS contain 138 character, 27 words and 3 sentences.



```
sns.heatmap(df.corr(),annot=True);
```

a. No_of_Characters has more positive corelation with No_Of_Word.
b. we don't have any negative corelations in the data.

- **Data Sources and their formats**

The SMS Spam Collection is a set of SMS tagged messages that have been collected for SMS Spam research. It contains one set of SMS messages in English of 5,574 messages, tagged according being ham (legitimate) or spam

The data set includes:

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' (Non spam) 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.

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.

This corpus has been collected from free or free for research sources at the Internet:

-> 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. 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.

- **Data Pre-processing**

I imported all the required libraries for cleansing the data.

```
#Importing Required libraries
import nltk
import re
import string
from nltk.corpus import stopwords
from wordcloud import WordCloud
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer
from sklearn.feature_extraction.text import TfidfVectorizer
```

After importing all the required libraries, I have defined stopwords and lemmatize to a variable.

```
from nltk.corpus import stopwords
def clean_text(df, df_column_name):

    #Converting all messages to lowercase
    df[df_column_name] = df[df_column_name].str.lower()

    #Replace email addresses with 'email'
    df[df_column_name] = df[df_column_name].str.replace(r'^.+@[^\.].*\.[a-z]{2,}$','emailaddress')

    #Replace URLs with 'webaddress'
    df[df_column_name] = df[df_column_name].str.replace(r'^http\://[a-zA-Z0-9\-\.]+\.[a-zA-Z]{2,3}(/\S*)?$','webaddress')

    #Replace money symbols with 'dollars' (£ can by typed with ALT key + 156)
    df[df_column_name] = df[df_column_name].str.replace(r'£|\$', 'dollars')

    #Replace 10 digit phone numbers (formats include paranthesis, spaces, no spaces, dashes) with 'phonenumber'
    df[df_column_name] = df[df_column_name].str.replace(r'^\(?[\d]{3}\)?[\s-]?[\d]{3}[\s-]?[\d]{4}$','phonenumber')

    #Replace numbers with 'numbr'
    df[df_column_name] = df[df_column_name].str.replace(r'\d+(\.\d+)?', 'numbr')

    #Remove punctuation
    df[df_column_name] = df[df_column_name].str.replace(r'[^\w\d\s]', ' ')

    #Replace whitespace between terms with a single space
    df[df_column_name] = df[df_column_name].str.replace(r'\s+', ' ')

    #Remove leading and trailing whitespace
    df[df_column_name] = df[df_column_name].str.replace(r'^\s+|\s+?$', '')

    #Remove stopwords
    stop_words = set(stopwords.words('english') + ['u', 'ü', 'â', 'ur', '4', '2', 'im', 'dont', 'doin', 'ure'])
    df[df_column_name] = df[df_column_name].apply(lambda x: ' '.join(term for term in x.split() if term not in stop_words))
```

Post on creating a function I have passed my data into the same to clean it.

```
#Calling the class
clean_text(df, 'SMS')
df['SMS'].head()
```

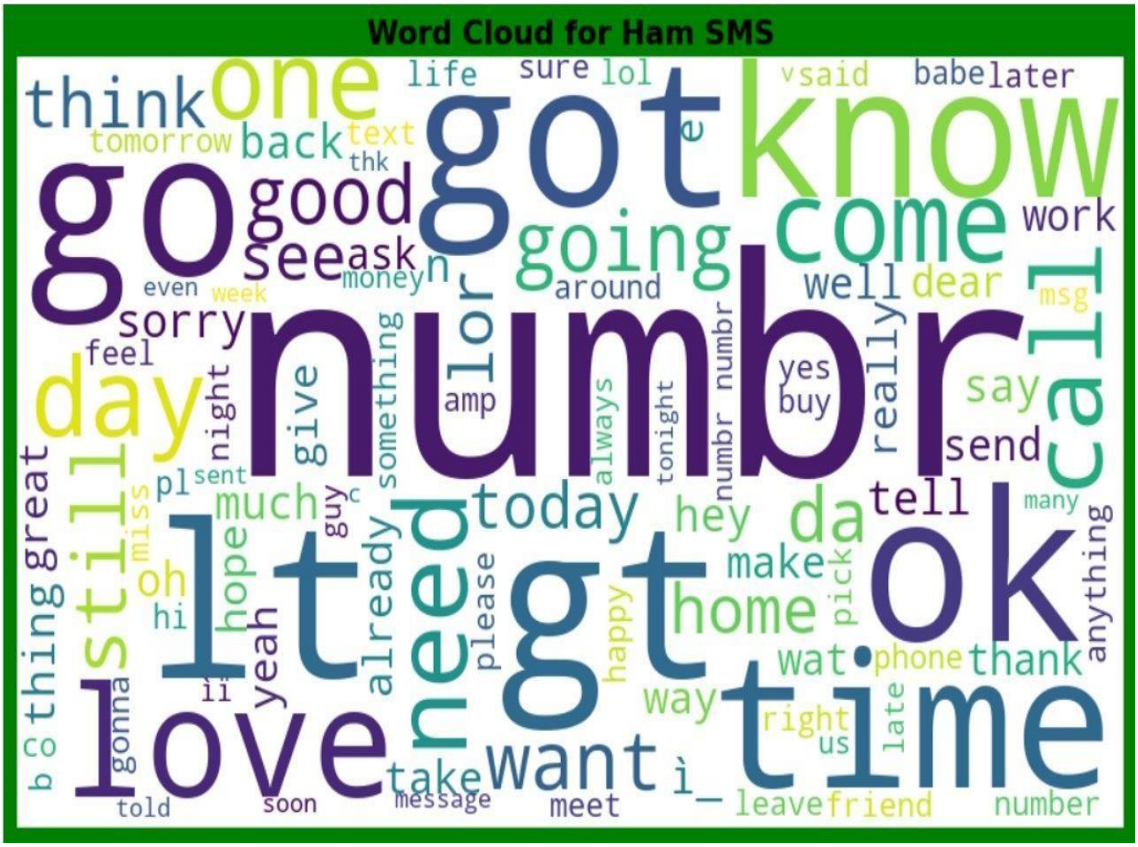
```
0    go jurong point crazy available bugis n great ...
1                                ok lar joking wif oni
2    free entry numbr wkly comp win fa cup final tk...
3                  dun say early hor c already say
4             nah think goes usf lives around though
Name: SMS, dtype: object
```

The total amount of data that is cleansed from the original data is 5572. Now the data is cleansed and ready for training but before which I converted the data into vectors for the machine learning models to understand the data, so I imported TFIDF vectorizer and I have made the max feature as 3000.

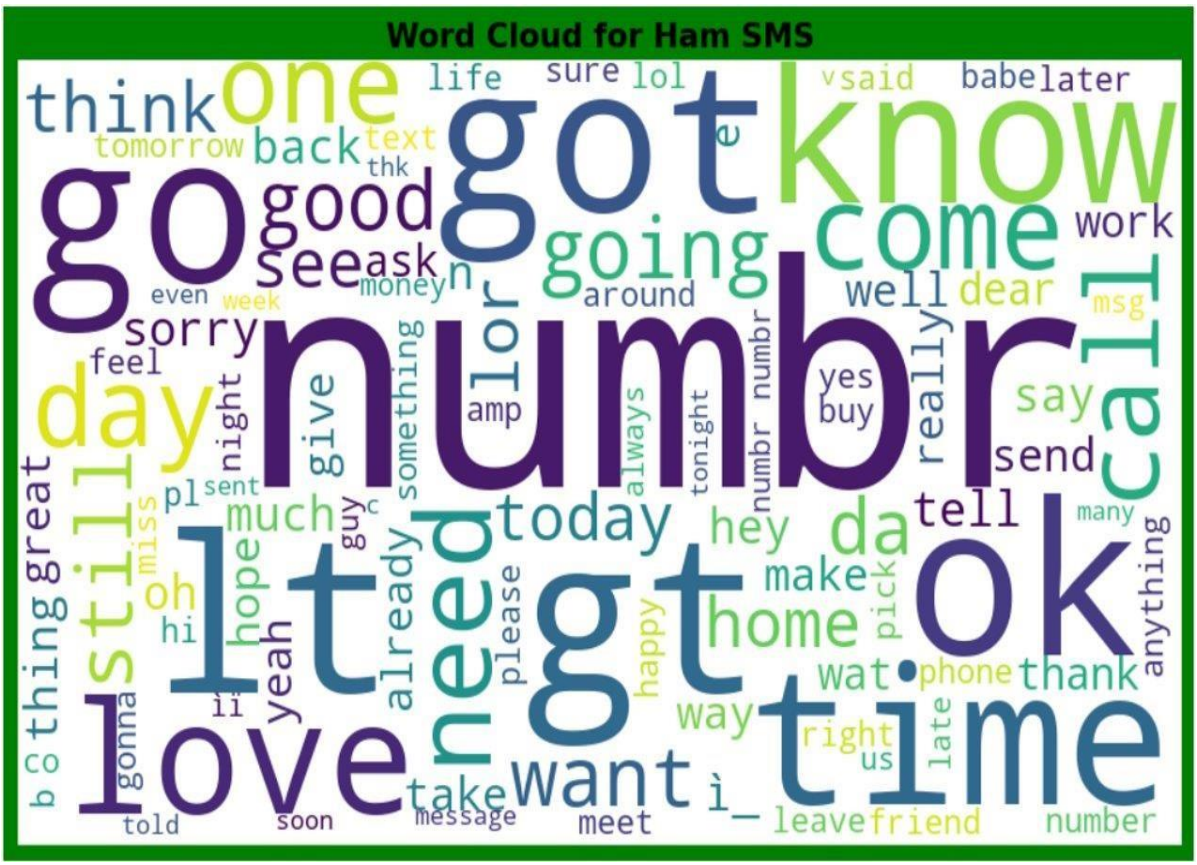
```
from sklearn.feature_extraction.text import TfidfVectorizer
tf = TfidfVectorizer(max_features=3000)
fertures = tf.fit_transform(df['SMS'])
X=fertures
Y=df[['Target']]
```

- **Data Inputs- Logic- Output Relationships**

I have analysed the input output logic with word cloud and I have word clouded the sentenced that as classified as foul language in every category.



We can see the foul words that are mostly used in Hum SMS classified sentences we are seeing top 400 words the words which are bigger in size are mostly used.



We can see the foul words that are mostly used in SMS classified sentences we are seeing top 400 words the words which are bigger in size are mostly used.

- **Hardware and Software Requirements and Tools Used**
  1. Python 3.10
  2. NumPy.
  3. Pandas.
  4. Matplotlib.
  5. Seaborn. 6. Data science.
  6. SciPy
  7. Sklearn.
  8. Anaconda Environment, Jupyter Notebook.

Model/s Development and Evaluation

- **Testing of Identified Approaches (Algorithms)**

I have started the training in selecting the best random state parameter for the model as follows.

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
maxAccu = 0
maxRs = 0
for i in range(1,200):
    X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size=.30,random_state=i)
    RF = RandomForestClassifier()
    RF.fit(X_train,Y_train)
    predRF = RF.predict(X_test)
    acc = accuracy_score(Y_test,predRF)
    if acc>maxAccu:
        maxAccu=acc
        maxRs = i
print(f'Best Accuracy is {maxAccu} on Random_state {maxRs}')
```

```
Best Accuracy is 0.988394584139265 on Random_state 195
```

After selecting the best random state parameter, I have spitted the data into test and train with test size as 30 %. Again, I have imported the required libraries to import my ML algorithms.

- **Run and evaluate selected models**

Logistic Regression

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import r2_score,confusion_matrix,classification_report,mean_absolute_error,mean_squared_error

LOR = LogisticRegression()
LOR.fit(X_train,Y_train)

# Prediction
predLOR = LOR.predict(X_test)
print('R2 Score :',r2_score(Y_test,predLOR))

# Mean Absolute Error(MAE)
print('Mean Absolute Error(MAE)',mean_absolute_error(Y_test,predLOR))

# Mean Squared Error(MSE)
print('Mean Squared Error',mean_squared_error(Y_test,predLOR))

# Root Mean Squared Error (RMSE)
print('Root Mean Squared Error',np.sqrt(mean_squared_error(Y_test,predLOR)))

print("-----------------------------------------------------")
# Accuracy Score
print(accuracy_score(Y_test, predLOR))
print("-----------------------------------------------------")
# Confusion Matrix
print(confusion_matrix(Y_test, predLOR))
print("-----------------------------------------------------")
# Classification Report
print(classification_report(Y_test,predLOR))
```

```
R2 Score : 0.7436475142447045
Mean Absolute Error(MAE) 0.024500322372662798
Mean Squared Error 0.024500322372662798
Root Mean Squared Error 0.1565257882032951
--------------------------------------------------
0.9754996776273372
--------------------------------------------------
[[1380    5]
 [  33  133]]
--------------------------------------------------
              precision    recall  f1-score   support

           0       0.98      1.00      0.99      1385
           1       0.96      0.80      0.88       166

    accuracy                           0.98      1551
   macro avg       0.97      0.90      0.93      1551
weighted avg       0.98      0.98      0.97      1551
```

Random Forest Classifier

```
from sklearn.ensemble import RandomForestClassifier

# Checking accuracy for Random Forest Classifier
RFC = RandomForestClassifier()
RFC.fit(X_train,Y_train)

# [Prediction]
predRFC = RFC.predict(X_test)
print('R2 Score:',r2_score(Y_test,predRFC))

# Mean Absolute Error(MAE)
print('Mean Absolute Error',mean_absolute_error(Y_test,predRFC))

# Mean Squared Error(MSE)
print('Mean Squared Error',mean_squared_error(Y_test,predRFC))

# Root Mean Squared Error (RMSE)
print('Root Mean Squared Error',np.sqrt(mean_squared_error(Y_test,predRFC)))
print("--------------------------------------------------")
# Accuracy Score
print('Accuracy Score: ',accuracy_score(Y_test, predRFC))
print("--------------------------------------------------")
# Confusion Matrix
print('Confusion Matrix:\n',confusion_matrix(Y_test, predRFC))
print("--------------------------------------------------")
# Classification Report
print(classification_report(Y_test,predRFC))
```

```
R2 Score: 0.8853159932147362
Mean Absolute Error 0.01096067053513862
Mean Squared Error 0.01096067053513862
Root Mean Squared Error 0.10469322105627765
-----------------------------------------------------
Accuracy Score:  0.9890393294648614
-----------------------------------------------------
Confusion Matrix:
 [[1385    0]
 [  17  149]]
-----------------------------------------------------
              precision    recall  f1-score   support

           0       0.99      1.00      0.99      1385
           1       1.00      0.90      0.95       166

    accuracy                           0.99      1551
   macro avg       0.99      0.95      0.97      1551
weighted avg       0.99      0.99      0.99      1551
```

MultinomialNB Classifier

```
from sklearn.naive_bayes import MultinomialNB

# Checking accuracy for MultinomialNB Classifier
MNB = MultinomialNB()
MNB.fit(X_train,Y_train)

# [Prediction]
predMNB = MNB.predict(X_test)
print('R2 Score:',r2_score(Y_test,predMNB))

# Mean Absolute Error(MAE)
print('Mean Absolute Error',mean_absolute_error(Y_test,predMNB))

# Mean Squared Error(MSE)
print('Mean Squared Error',mean_squared_error(Y_test,predMNB))

# Root Mean Squared Error (RMSE)
print('Root Mean Squared Error',np.sqrt(mean_squared_error(Y_test,predMNB)))
print("-----------------------------------------------------")
# Accuracy Score
print('Accuracy Score: ',accuracy_score(Y_test, predMNB))
print("-----------------------------------------------------")
# Confusion Matrix
print('Confusion Matrix:\n',confusion_matrix(Y_test, predMNB))
print("-----------------------------------------------------")
# Classification Report
print(classification_report(Y_test,predMNB))
```

```
R2 Score: 0.8448392849375843
Mean Absolute Error 0.014829142488716958
Mean Squared Error 0.014829142488716958
Root Mean Squared Error 0.12177496659296178
----------------------------------------------------
Accuracy Score:  0.9851708575112831
----------------------------------------------------
Confusion Matrix:
 [[1385    0]
 [  23  143]]
----------------------------------------------------
              precision    recall  f1-score   support

           0       0.98      1.00      0.99      1385
           1       1.00      0.86      0.93       166

    accuracy                           0.99      1551
   macro avg       0.99      0.93      0.96      1551
weighted avg       0.99      0.99      0.98      1551
```

BernoulliNB

```python
from sklearn.naive_bayes import BernoulliNB

# Checking accuracy for BernoulliNB Classifier
BNB = BernoulliNB()
BNB.fit(X_train,Y_train)

# [Prediction]
predBNB = BNB.predict(X_test)
print('R2 Score:',r2_score(Y_test,predBNB))

# Mean Absolute Error(MAE)
print('Mean Absolute Error',mean_absolute_error(Y_test,predBNB))

# Mean Squared Error(MSE)
print('Mean Squared Error',mean_squared_error(Y_test,predBNB))

# Root Mean Squared Error (RMSE)
print('Root Mean Squared Error',np.sqrt(mean_squared_error(Y_test,predBNB)))
print("----------------------------------------------------")
# Accuracy Score
print('Accuracy Score: ',accuracy_score(Y_test, predBNB))
print("----------------------------------------------------")
# Confusion Matrix
print('Confusion Matrix:\n',confusion_matrix(Y_test, predBNB))
print("----------------------------------------------------")
# Classification Report
print(classification_report(Y_test,predBNB))
```

```
R2 Score: 0.8583315210299682
Mean Absolute Error 0.013539651837524178
Mean Squared Error 0.013539651837524178
Root Mean Squared Error 0.11636000961466177
----------------------------------------------------
Accuracy Score:  0.9864603481624759
----------------------------------------------------
Confusion Matrix:
 [[1379    6]
 [  15  151]]
----------------------------------------------------
              precision    recall  f1-score   support

           0       0.99      1.00      0.99      1385
           1       0.96      0.91      0.93       166

    accuracy                           0.99      1551
   macro avg       0.98      0.95      0.96      1551
weighted avg       0.99      0.99      0.99      1551
```

Extra Trees Classifier

```python
from sklearn.ensemble import ExtraTreesClassifier

# Checking accuracy for Support Vector Machine Classifier
ETC = ExtraTreesClassifier()
ETC.fit(X_train,Y_train)

# [Prediction]
predETC = ETC.predict(X_test)
print('R2 Score:',r2_score(Y_test,predETC))

# Mean Absolute Error(MAE)
print('Mean Absolute Error',mean_absolute_error(Y_test,predETC))

# Mean Squared Error(MSE)
print('Mean Squared Error',mean_squared_error(Y_test,predETC))

# Root Mean Squared Error (RMSE)
print('Root Mean Squared Error',np.sqrt(mean_squared_error(Y_test,predETC)))
print("----------------------------------------------------")
# Accuracy Score
print('Accuracy Score: ',accuracy_score(Y_test, predETC))
print("----------------------------------------------------")
# Confusion Matrix
print('Confusion Matrix:\n',confusion_matrix(Y_test, predETC))
print("----------------------------------------------------")
# Classification Report
print(classification_report(Y_test,predETC))
```

```
R2 Score: 0.8583315210299682
Mean Absolute Error 0.013539651837524178
Mean Squared Error 0.013539651837524178
Root Mean Squared Error 0.11636000961466177
----------------------------------------------------
Accuracy Score:  0.9864603481624759
----------------------------------------------------
Confusion Matrix:
 [[1382    3]
 [  18  148]]
----------------------------------------------------
              precision    recall  f1-score   support

           0       0.99      1.00      0.99      1385
           1       0.98      0.89      0.93       166

    accuracy                           0.99      1551
   macro avg       0.98      0.94      0.96      1551
weighted avg       0.99      0.99      0.99      1551
```

AdaBoostClassifier

```python
from sklearn.ensemble import AdaBoostClassifier
ADA = AdaBoostClassifier()
ADA.fit(X_train,Y_train)

# [Prediction]
predADA = ADA.predict(X_test)
print('R2 Score:',r2_score(Y_test,predADA))

# Mean Absolute Error(MAE)
print('Mean Absolute Error',mean_absolute_error(Y_test,predADA))

# Mean Squared Error(MSE)
print('Mean Squared Error',mean_squared_error(Y_test,predADA))

# Root Mean Squared Error (RMSE)
print('Root Mean Squared Error',np.sqrt(mean_squared_error(Y_test,predADA)))
print("----------------------------------------------------")
# Accuracy Score
print('Accuracy Score: ',accuracy_score(Y_test, predADA))
print("----------------------------------------------------")
# Confusion Matrix
print('Confusion Matrix:\n',confusion_matrix(Y_test, predADA))
print("----------------------------------------------------")
# Classification Report
print(classification_report(Y_test,predADA))
```

```
R2 Score: 0.8111086947066244
Mean Absolute Error 0.018052869116698903
Mean Squared Error 0.018052869116698903
Root Mean Squared Error 0.13436096574786482
----------------------------------------------------
Accuracy Score:  0.9819471308833011
----------------------------------------------------
Confusion Matrix:
 [[1378    7]
 [  21  145]]
----------------------------------------------------
              precision    recall  f1-score   support

           0       0.98      0.99      0.99      1385
           1       0.95      0.87      0.91       166

    accuracy                           0.98      1551
   macro avg       0.97      0.93      0.95      1551
weighted avg       0.98      0.98      0.98      1551
```

Support Vector Machine Classifier

```
from sklearn.svm import SVC

# Checking accuracy for Support Vector Machine Classifier
svc = SVC()
svc.fit(X_train,Y_train)

# [Prediction]
predsvc = svc.predict(X_test)
print('R2 Score:',r2_score(Y_test,predsvc))

# Mean Absolute Error(MAE)
print('Mean Absolute Error',mean_absolute_error(Y_test,predsvc))

# Mean Squared Error(MSE)
print('Mean Squared Error',mean_squared_error(Y_test,predsvc))

# Root Mean Squared Error (RMSE)
print('Root Mean Squared Error',np.sqrt(mean_squared_error(Y_test,predsvc)))
print("----------------------------------------------------")
# Accuracy Score
print('Accuracy Score: ',accuracy_score(Y_test, predsvc))
print("----------------------------------------------------")
# Confusion Matrix
print('Confusion Matrix:\n',confusion_matrix(Y_test, predsvc))
print("----------------------------------------------------")
# Classification Report
print(classification_report(Y_test,predsvc))
```

```
R2 Score: 0.8650776390761603
Mean Absolute Error 0.012894906511927788
Mean Squared Error 0.012894906511927788
Root Mean Squared Error 0.11355574187124043
--------------------------------------------------------
Accuracy Score:  0.9871050934880722
--------------------------------------------------------
Confusion Matrix:
 [[1385    0]
 [  20  146]]
--------------------------------------------------------
```

	precision	recall	f1-score	support
0	0.99	1.00	0.99	1385
1	1.00	0.88	0.94	166
accuracy			0.99	1551
macro avg	0.99	0.94	0.96	1551
weighted avg	0.99	0.99	0.99	1551

- **Key-Metrics for success in solving problem under consideration.**

```python
from sklearn.model_selection import cross_val_score

#cv score for Logistic Regression
print('Logistic Regression',cross_val_score(LOR,X,Y,cv=5).mean())

# cv score for Random Forest Classifier
print('Random Forest Classifier',cross_val_score(RFC,X,Y,cv=5).mean())

# cv score for KNeighbors Classifier
print('BernoulliNB Classifier:',cross_val_score(BNB,X,Y,cv=5).mean())

# cv score for Support Vector  Classifier
print('Support Vector  Classifier',cross_val_score(svc,X,Y,cv=5).mean())

# cv score for Extra Trees Classifier
print('Extra Trees Classifier:',cross_val_score(ETC,X,Y,cv=5).mean())

# cv score for Naive Bias Classifier
print('MultinomialNB Classifier:',cross_val_score(MNB,X,Y,cv=5).mean())

# cv score for AdaBoosting Classifier
print('AdaBoosting Classifier:',cross_val_score(ADA,X,Y,cv=5).mean())
```

```
Logistic Regression 0.968466336242489
Random Forest Classifier 0.9820080477698241
BernoulliNB Classifier: 0.9854898597725728
Support Vector  Classifier 0.982589254785502
Extra Trees Classifier: 0.9814272152432025
MultinomialNB Classifier: 0.9798800137062994
AdaBoosting Classifier: 0.9754310837151563
```

CONCLUSION

- ## Key Findings and Conclusions of the Study

  The finding of the study is that only few users over online use unparliamentary language. And most of these sentences have more stop words, and are being long. As discussed before few motivated disrespectful crowds uses these foul languages in the online forum to bully the people around and to stop them from doing the things that they are supposed to do. Our Study helps the online forms and social media to induce a ban to profanity or usage of profanity over these forms.

- ## Learning Outcomes of the Study in respect of Data Science

  The use of social media is the most common trend among the activities of today's people. Social networking sites offer today's teenagers a platform for communication and entertainment. They use social media to collect more information from their friends and followers. The vastness of social media sites ensures that not all of them provide a decent environment for children. In such cases, the impact of the negative influences of social media on teenage users increases with an increase in the use of **offensive language** in social conversations. This increase could lead to **frustration**, **depression** and a large change in their behaviour. Hence, I propose a novel approach to classify bad language usage in text conversations. I have considered the English medium for textual conversation. I have developed our system based on a foul language classification approach; it is based on an improved version of a Random Forest Classification Algorithm that detects offensive language usage in a conversation. As per our evaluation, we found that lesser number of users conversation is not decent all the time. We trained 5572 observations for eight context categories using a Random Forest algorithm for context detection. Then, the system classifies the use of foul language in one of the trained contexts in the text conversation. In our testbed, we observed 10% of participants used foul language during their text conversation. Hence, our proposed approach can identify the impact of foul

language in text conversations using a classification technique and emotion detection to identify the foul language usage

- ## Limitations of this work and Scope for Future Work

  The limitation of the study is that we have an imbalanced data so our model learnt more about the non-abusive sentence more than the abusive sentence. Which makes our model act like an overfit model when tested with live data. And also, model tend to not identify a foul or a sarcastically foul language.