EMAIL SPAM CLASSIFIER

By : Wesley Sirra

AGENDA.

- Overview
- Problem Statement.
- Problem Understanding.
- Importance of Malignant Comments Classification.
- Exploratory Data Analysis (Steps).
- Visualizations.
- Word Clouds.
- Data Analysis Steps.
- Model Building.
- Analysis of Models.
- Cross Validation Scores.
- Hyper Parameter Tuning and Creating the Final Model.
- Saving the model and predicting the results.
- Conclusion.

OVERVIEW.

In this particular presentation we will be looking at:

- How to analyze the dataset of SMS SPAM CLASSIFIER.
- What are the EDA steps in cleaning the dataset.
- Overall analysis on the problem.
- Model building from the cleaned dataset.
- Predictions for test dataset from saved model.

Problem STATEMENT.

In today's globalized world, email is a primary source of communication. This communication can vary from personal, business, corporate to government. With the rapid increase in email usage, there has also been increase in the SPAM emails. SPAM emails, also known as junk email involves nearly identical messages sent to numerous recipients by email. Apart from being annoying, spam emails can also pose a security threat to computer system. It is estimated that spam cost businesses on the order of $100 billion in 2007. In this project, we use text mining to perform automatic spam filtering to use emails effectively. We try to identify patterns using Data-mining classification algorithms to enable us classify the emails as HAM or SPAM.

Problem STATEMENT.

- At least 97% of American use text messages over mobile phones every day. In 2016, according to the research conducted by Portio research, 8.3 trillion messages exchanged over the mobile phones. The rising flood of big data shows an exchange of 23 billion messages per day and 16 million messages per minute. There are around 6.4 billion mobile subscribers around the world by the end of 2012. According to Portio Research, there will be a CAGR growth of 4.8% of growth in mobile subscriber base from 2014 to 2017. By the end of 2017, the mobile subscriber reached to 7.4 billion mobile subscribers. The proliferation of smart devices powered by exponential computing has shown a significant rise in the global smartphone system-on-chip market lead by Qualcomm, Apple, MediaTrek, Samsung, HiSilicon, Spreadtrum, and a vast number of other smartphone chip manufacturers in the market.

Problem UNDERSTANDING.

In the past few years, it is seen that the cases related to social media hatred have increased exponentially. The social media is turning into a dark venomous pit for people now a days. Online hate is the result of difference in opinion, race, religion, occupation, nationality etc. In social media the people spreading or involved in such kind of activities uses filthy languages, aggression, images etc. to offend and gravely hurt the person on the other side. This is one of the major concerns now.

The result of such activities can be dangerous. It gives mental trauma to the victims making their lives miserable. People who are not well aware of mental health online hate or cyber bullying become life threatening for them. Such cases are also at rise. It is also taking its toll on religions. Each and every day we can see an incident of fighting between people of different communities or religions due to offensive social media posts.

Importance of SMS SPAM CLASSIFIER.

Every day, we get a tremendous amount of short content data from the blast of online correspondence, web-based business and the utilization of advanced gadgets. This volume of data requires text mining apparatuses to carry out the various report tasks in an opportune and suitable way. Detecting and controlling verbal AND fake abuse in an automated fashion is inherently an NLP task (Natural Language Processing). Text Classification is a great point for NLP.

Nowadays, every email and short messaging service and applications use machine learning approach. Machine Learning has simplified the task that may take long duration to complete without it. Most of the approaches require text analysis and classification techniques. Classification of the comments is necessary before posting on online platforms. This classification model helps to prevent the online abuse and cyber bullying.

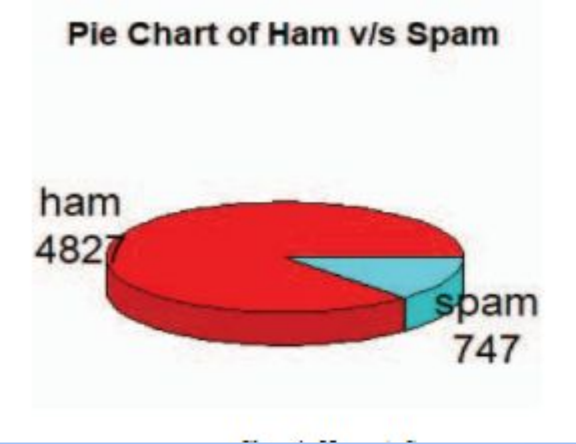
Exploratory Data Analysis.

- Importing necessary libraries and importing the Train & Test datasets.
- Checked some statistical information like shape, number of unique values present, info, finding zero values etc on both the datasets.
- Checked for null values and did not find any null values In both datasets. And removed Id.
- Conducted some feature engineering and created new columns via label: which contain both good and bad comments which is the sum of all the labels, comment length: which contains the length of comment text.
- Visualized each feature using seaborn and matplotlib libraries by plotting categorical plots like pie plot, count plot, distribution plot and word cloud for each label.

Exploratory Data Analysis.

- Done text pre-processing techniques like Removing Punctuations and other special characters, Splitting the comments into individual words, Removing Stop Words, Stemming and Lemmatization.
- Then created new column as clean _length after cleaning the data.
- All these steps were done on both train and test datasets.
- Checked correlation using heatmap.
- After getting a cleaned data used TF-IDF vectorizer.
- Lastly, proceeded with model building.

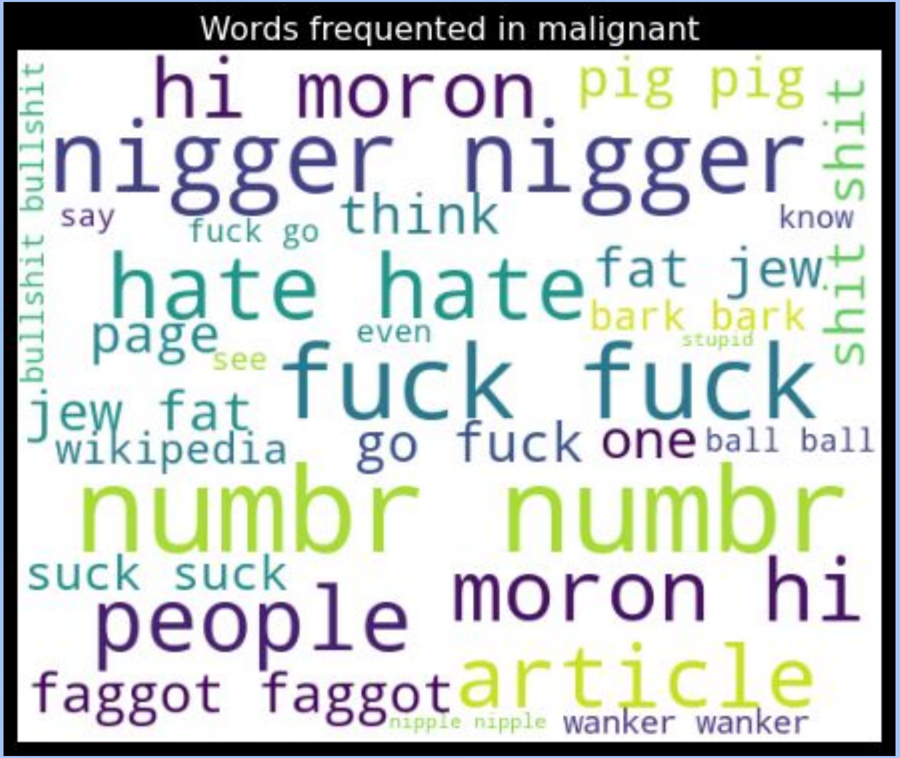
VISUALIZATIONS.



OBSERVATIONS:

From the pie chart we can notice approximately 4827 of the MESSAGE are SPAM, 747 of the MESSEAGE are rude and are abuse. The count of SPAM are high compared to other type of MESSAGE and the count of threat comments are very less.

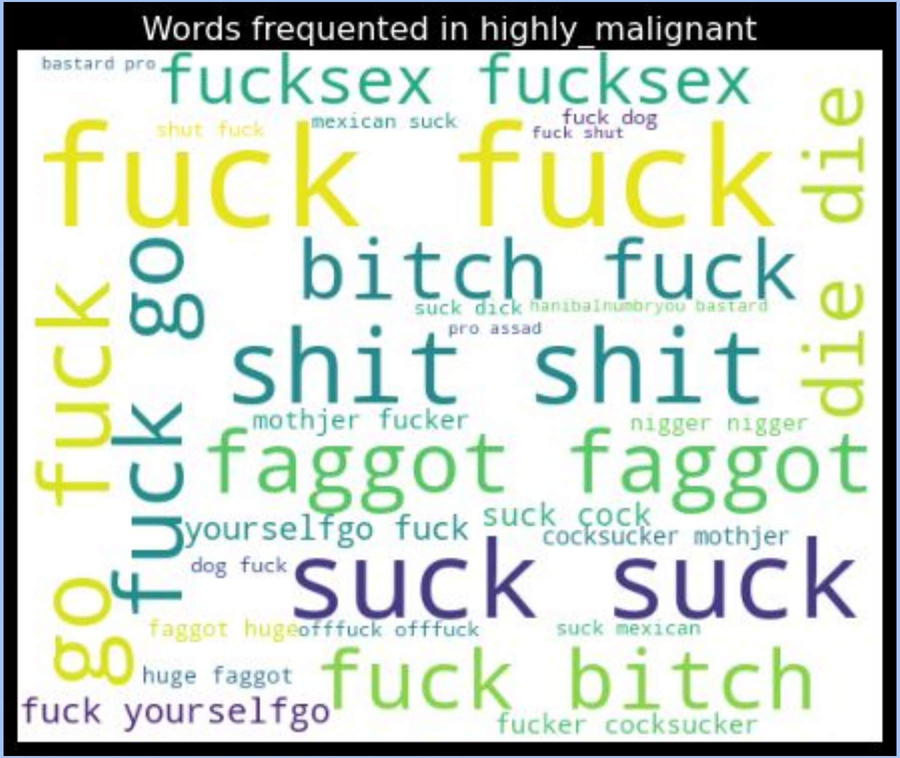
Word Clouds.



<u>OBSERVATIONS</u>:

These are the toxic words which frequently appear in the Malignant column.

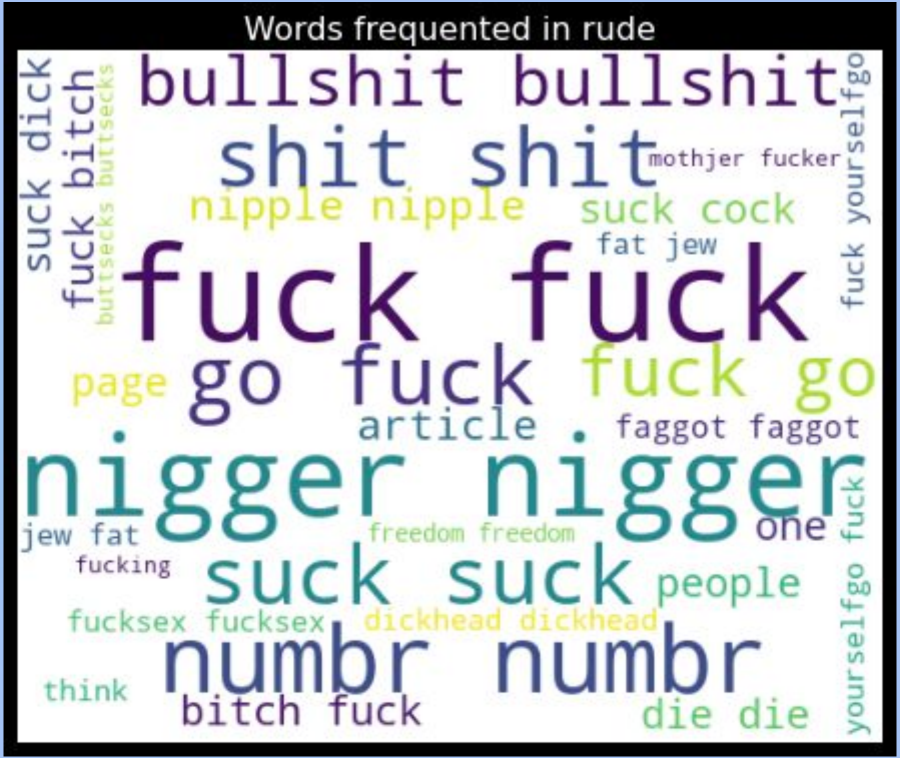
Word Clouds.



OBSERVATIONS:

These are the toxic words which frequently appear in the Highly Malignant column.

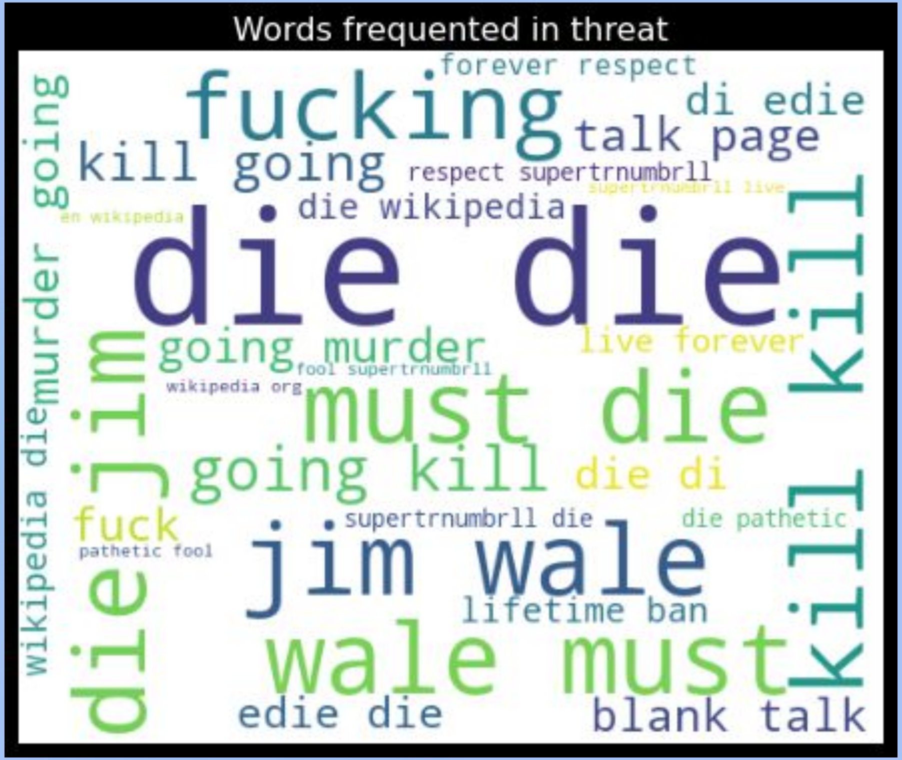
WORD CLOUDS.



OBSERVATIONS:

These are the toxic words which frequently appear in the rude column.

Word Clouds.



Words frequented in threat

OBSERVATIONS:

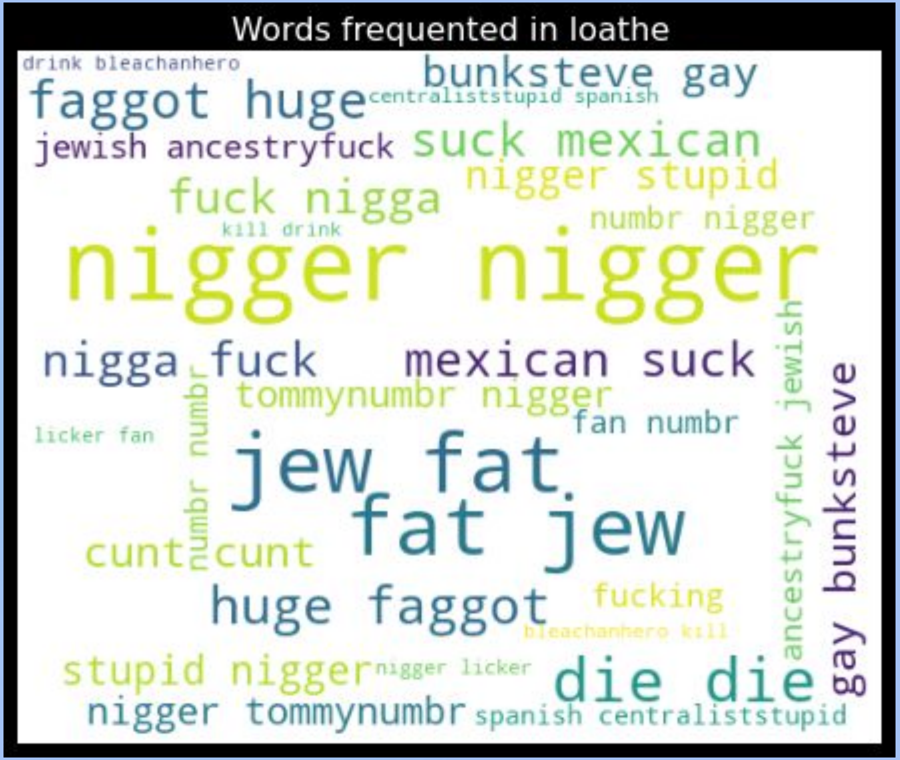
These are the toxic words which frequently appear in the threat column.

Word Clouds.

<u>OBSERVATIONS</u>:

These are the toxic words which frequently appear in the abuse column.

Word Clouds.



<u>OBSERVATIONS</u>:

These are the toxic words which frequently appear in the loathe column.

DATA ANALYSIS STEPS.

- I have extracted some features and removed the feature “Id” to improve data normality and linearity.
- Done text pre-processing techniques like: Removing Punctuations and other special characters, Splitting the comments into individual words, Removing Stop Words, Stemming and Lemmatization.
- Then created new column as clean_length after cleaning the data.
- All these steps were done on both train and test datasets.
- Used Pearson’s correlation coefficient and heat map to check the correlation.

DATA ANALYSIS STEPS.

- After getting a cleaned data used TF-IDF vectorizer. It'll help to transform the text data to feature vector which can be used as input in our modelling.
- Balanced the data using Random-oversampler mechanism.
- Split train and test to build machine learning models.
- Model building process will be shown in the further steps.

MODEL BUILDING.

In this project there were 6 features which defines the type of comment like malignant, hate, abuse, threat, loathe but we created another feature named as “label” which is combined of all the above features and contains the labeled data into the format of 0 and 1 where 0 represents “NO” and 1 represents “Yes”.

In this NLP based project we need to predict the multiple labels which are binary. I have converted text into feature vectors using TF-IDF vectorizer and separated our features and labels. Also, before building the model, I made sure that the input data was cleaned and scaled before it was fed into the machine learning models.

After the pre-processing and data cleaning I used remaining independent features for model building and prediction.

MODEL BUILDING.

The classification algorithms used on training the data are as follows:

1.gnb = GaussianNB()

2. mnb = MultinomialNB()

3.bnb = BernoulliNB()

4. ADABOOST CLASSIFIER MODEL.

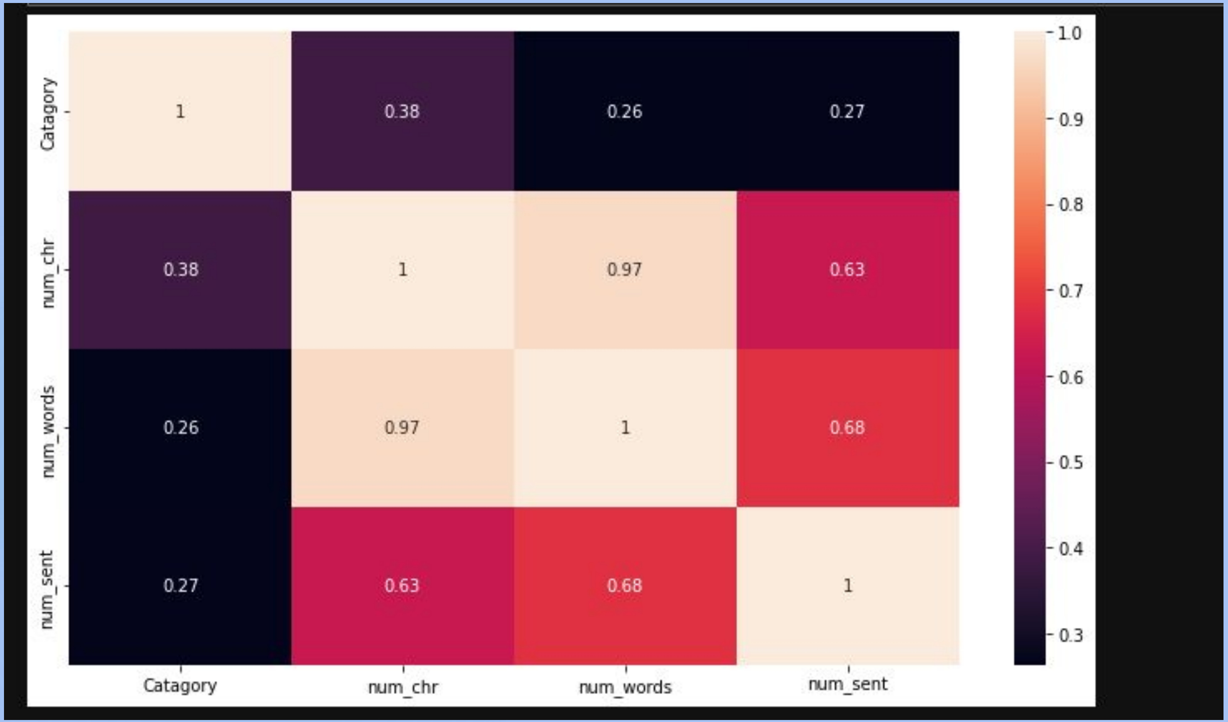
GAUSSIAN NB

The GAUSSIAN NB CLASSIFIER Model gave us an accuracy score of 86.46 %.

```
gnb.fit(A_train,B_train)
b_pred1 = gnb.predict(A_test)
print(accuracy_score(B_test,b_pred1))
print(confusion_matrix(B_test,b_pred1))
print(precision_score(B_test,b_pred1))
```

```
0.8694390715667312
[[788 108]
 [ 27 111]]
0.5068493150684932
```

HEAT MAP


Catagory
num_chr
num_words
num_sent
1
0.38
0.26
0.27
0.38
1
0.97
0.63
0.26
0.97
1
0.68
0.27
0.63
0.68
1
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3

MULTINOMIAL NB CLASSIFIER

The MULTINOMIAL NB CLASSIFIER Model gave us an accuracy score of 97.08 %.

```
mnb.fit(A_train,B_train)
b_pred2 = mnb.predict(A_test)
print(accuracy_score(B_test,b_pred2))
print(confusion_matrix(B_test,b_pred2))
print(precision_score(B_test,b_pred2))
```

```
0.9709864603481625
[[896   0]
 [ 30 108]]
1.0
```

BERNOULI NB CLASSIFIER

The BERNOULI NB CLASSIFIER gave us an accuracy score of 98.35 %.

```
bnb.fit(A_train,B_train)
b_pred3 = bnb.predict(A_test)
print(accuracy_score(B_test,b_pred3))
print(confusion_matrix(B_test,b_pred3))
print(precision_score(B_test,b_pred3))
```

```
0.9835589941972921
[[895   1]
 [ 16 122]]
0.991869918699187
```

ADABOOST CLASSIFIER MODEL.

The ADA Boost CLASSIFIER Model gave us an accuracy score of 92.68 %.

```
In [79]: from sklearn.ensemble import AdaBoostClassifier

         abc=AdaBoostClassifier()
         abc.fit(train_x,train_y)
         pred_abc=abc.predict(x_test)

         print("Accuracy score: ",accuracy_score(y_test,pred_abc))
         print("Roc_auc_score: ",roc_auc_score(y_test,pred_abc))
         print("Log loss : ",log_loss(y_test,pred_abc))
         print("Hamming loss: ",hamming_loss(y_test,pred_abc))
         print('Confusion matrix: \n',confusion_matrix(y_test,pred_abc))
         print('Classification Report:\n ',classification_report(y_test,pred_abc))

         Accuracy score:  0.9268465909090909
         Roc_auc_score:  0.8144580882846922
         Log loss :  2.52666117490942
         Hamming loss:  0.07315340909090909
         Confusion matrix:
          [[41092  1912]
          [ 1590  3278]]
         Classification Report:
                        precision    recall  f1-score   support

                    0       0.96      0.96      0.96     43004
                    1       0.63      0.67      0.65      4868

             accuracy                           0.93     47872
            macro avg       0.80      0.81      0.81     47872
         weighted avg       0.93      0.93      0.93     47872
```

XGBOOST CLASSIFIER MODEL.

The XG Boost CLASSIFIER Model gave us an accuracy score of 94.89 %.

In [80]:
```
from xgboost import XGBClassifier

xgb=XGBClassifier()
xgb.fit(train_x,train_y)
pred_xgb=xgb.predict(x_test)

print("Accuracy score: ",accuracy_score(y_test,pred_xgb))
print("Roc_auc_score: ",roc_auc_score(y_test,pred_xgb))
print("Log loss : ",log_loss(y_test,pred_xgb))
print("Hamming loss: ",hamming_loss(y_test,pred_xgb))
print('Confusion matrix: \n',confusion_matrix(y_test,pred_xgb))
print('Classification Report:\n ',classification_report(y_test,pred_xgb))
```

```
Accuracy score:  0.9489054144385026
Roc_auc_score:  0.8539703592954644
Log loss :  1.7647637574570403
Hamming loss:  0.051094585561497326
Confusion matrix:
 [[41849  1155]
 [ 1291  3577]]
Classification Report:
                precision    recall  f1-score   support

           0       0.97      0.97      0.97     43004
           1       0.76      0.73      0.75      4868

    accuracy                           0.95     47872
   macro avg       0.86      0.85      0.86     47872
weighted avg       0.95      0.95      0.95     47872
```

EXTRA TREES CLASSIFIER MODEL.

The Extra Trees CLASSIFIER Model gave us an accuracy score of 95.30 %.

```
In [81]: from sklearn.ensemble import ExtraTreesClassifier

         etc=ExtraTreesClassifier()
         etc.fit(train_x,train_y)
         pred_etc=etc.predict(x_test)

         print("Accuracy score: ",accuracy_score(y_test,pred_etc))
         print("Roc_auc_score: ",roc_auc_score(y_test,pred_etc))
         print("Log loss : ",log_loss(y_test,pred_etc))
         print("Hamming loss: ",hamming_loss(y_test,pred_etc))
         print('Confusion matrix: \n',confusion_matrix(y_test,pred_etc))
         print('Classification Report:\n ',classification_report(y_test,pred_etc))

         Accuracy score:  0.9530414438502673
         Roc_auc_score:  0.8075421238833759
         Log loss :  1.6218981192737882
         Hamming loss:  0.04695855614973262
         Confusion matrix:
          [[42582   422]
          [ 1826  3042]]
         Classification Report:
                        precision    recall  f1-score   support

                    0       0.96      0.99      0.97     43004
                    1       0.88      0.62      0.73      4868

             accuracy                           0.95     47872
            macro avg       0.92      0.81      0.85     47872
         weighted avg       0.95      0.95      0.95     47872
```

Cross Valldatlon Scores.

.

- The cross validation score of the Multinomial NB Classifier Model is 94.63 %.
- The cross validation score of the Ada boost classifier Model is 94.57 %.
- The cross validation score of the XG Boost Classifier Model is 95.36 %.
- The cross validation score of the Extra Trees Classifier Model is 95.62 %.

From the above Cross Validation Scores, the highest CV score belongs to the Linear SVC model, followed by the Extra Trees Classifier & Logistic Regression Model. Next the XG Boost Classifier model , the Multinomial NB Classifier and the Ada Boost Classifier Model. Lastly, the Decision Tree Classifier.

HYPER PARAMETER TUNING.

Since the Accuracy Score and the cross validation score of the MULTINOMIAL NB CLASSIFER Model are good and the AUC score is the highest among others we shall consider this model for hyper parameter tuning.

We shall use Grid SearchCV for hyper parameter tuning.

After multiple tries with hyper parameter tuning, the highest accuracy score obtained was 94.49 %.

HYPER PARAMETER TUNING.

```
In [120]: from sklearn.model_selection import GridSearchCV

In [121]: parameters={
              'C': [0.2,0.3,0.4],
              'penalty': ['l1', 'l2'],
              'solver':['newton-cg','lbfgs'],
              'multi_class':['auto','ovr']}
          grid_lg = GridSearchCV(lg, param_grid = parameters, cv = 4, scoring='accuracy')

In [122]: grid_lg.fit(train_x,train_y)

Out[122]: GridSearchCV(cv=4, estimator=LogisticRegression(),
                       param_grid={'C': [0.2, 0.3, 0.4], 'multi_class': ['auto', 'ovr'],
                                   'penalty': ['l1', 'l2'],
                                   'solver': ['newton-cg', 'lbfgs']},
                       scoring='accuracy')

In [123]: grid_lg.best_params_

Out[123]: {'C': 0.4, 'multi_class': 'auto', 'penalty': 'l2', 'solver': 'newton-cg'}
```

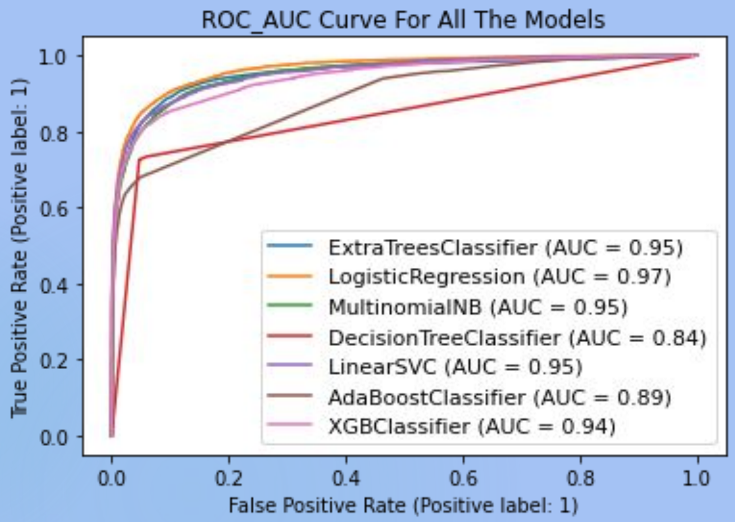
HYPER PARAMETER TUNING [FINAL MODEL].

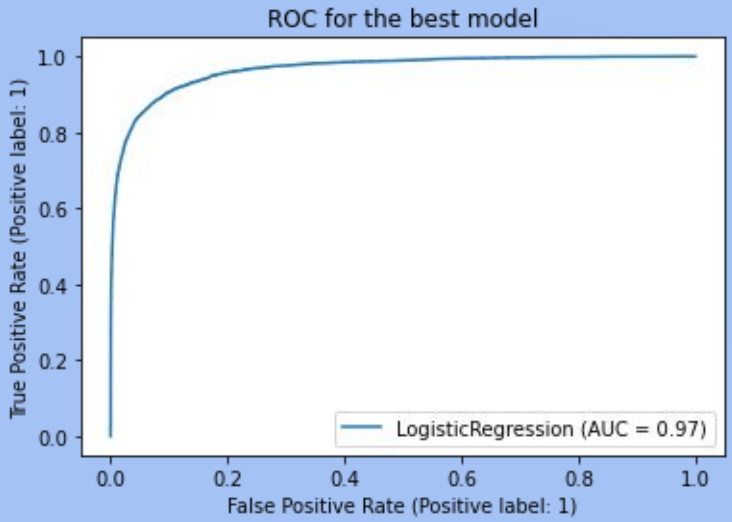
I have successfully incorporated hyper parameter tuning using best parameters of Logistic Regression and the accuracy of the model has been increased, We received the accuracy score as 94.49%, which is very good.

```
In [124]: Final_Model= LogisticRegression(C=0.4,penalty='l2',solver='newton-cg',multi_class='auto')

          Final_Model.fit(train_x,train_y)
          pred = Final_Model.predict(x_test)

          print("Accuracy score: ",accuracy_score(y_test,pred))
          print("Roc_auc_score: ",roc_auc_score(y_test,pred))
          print("Log loss : ",log_loss(y_test,pred))
          print("Hamming loss: ",hamming_loss(y_test,pred))
          print('Confusion matrix: \n',confusion_matrix(y_test,pred))
          print('Classification Report:\n ',classification_report(y_test,pred))
```

```
Accuracy score:  0.9449782754010695
Roc_auc_score:  0.894047844969343
Log loss :  1.9004132247817331
Hamming loss:  0.05502172459893048
Confusion matrix:
 [[41197  1807]
 [  827  4041]]
Classification Report:
               precision    recall  f1-score   support

           0       0.98      0.96      0.97     43004
           1       0.69      0.83      0.75      4868

    accuracy                           0.94     47872
   macro avg       0.84      0.89      0.86     47872
weighted avg       0.95      0.94      0.95     47872
```

ROC-AUC Curve.





I have generated the ROC Curve for all the models and for the best model and compared it with AUC. The AUC score for my final model was 97%.

Saving the model and predicting the results.

I have saved my final best model using joblib library in .pkl format, and loaded saved model for predictions for test data. Using classification model, we have got the predicted values for malignant comments classification.

```
In [73]: import pickle
         pickle.dump(tfidf,open('vectorizer.pkl','wb'))
         pickle.dump(mnb,open('model.pkl','wb'))
```

Saving the model and predicting the results.

In [130]:
```
# Predicting the values for test data after loading trained model
Predictions = model.predict(x1)
Predictions
```

Out[130]:
```
array([0, 0, 0, ..., 0, 0, 0])
```

In [131]:
```
# Adding the predicted values to test dataframe
test_mc['Predicted_Values']=Predictions
test_mc
```

Out[131]:

	id	comment_text	comment_length	clean_length	Predicted_Values
0	00001cee341fdb12	yo bitch ja rule succesful ever whats hating s...	367	227	0
1	0000247867823ef7	rfc title fine imo	50	18	0
2	00013b17ad220c46	source zawe ashton lapland	54	26	0
3	00017563c3f7919a	look back source information updated correct f...	205	109	0
4	00017695ad8997eb	anonymously edit article	41	24	0
...	...	...	...	...	...
153159	fffcd0960ee309b5	totally agree stuff nothing long crap	60	37	0
153160	fffd7a9a6eb32c16	throw field home plate get faster throwing cut...	198	107	0
153161	fffda9e8d6fafa9e	okinotorishima category see change agree corre...	423	238	0
153162	fffe8f1340a79fc2	one founding nation eu germany law return quit...	502	319	0
153163	ffffce3fb183ee80	stop already bullshit welcome fool think kind ...	141	74	0

153164 rows × 5 columns

CONCLUSION.

This project gives an idea of NLP text processing in machine learning. Apart from applying the techniques that we have learnt in the EDA, we also classified hate and offensive comments so that it can be controlled and restricted from spreading hatred and cyber bullying.

From this dataset we were able to understand the idea of Natural Language Processing using machine learning models. This model helps us to understand whether the online comments are malignant or non malignant.

We have mentioned step by step procedure to analyze the data and checked the correlation between label and feature.

CONCLUSION.

We got the Logistic Regression Model as the best model and performed hyper parameter tuning using the best parameters of Logistic Regression and plotted AUC-ROC score and the model accuracy and roc-auc score increased after tuning.

After that we saved the model in a pickle with a filename in order to use whenever we require. Then we loaded the saved file and predicted the values for test data. Further we saved the predicted values test data into a csv file.

THANK YOU