FLIP ROBO TECHNOLOGIES

EMAIL (SMS) SPAM CLASSIFIER

NAME- SHASHI SAHU

BATCH -30

SME- MOHD KASHIF SIR

ACKNOWLEDGEMENT

I would like to express my special gratitude to “Flip Robo Technologies” team, who has given me this opportunity to deal with a beautiful dataset related to spam list dataset and it has helped me to improve my analysation skills. And special thanks to our SME – Mohd Kashif Sir for his useful guidance.

INTRODUCTION

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.

DATA PREPROCESSING

The emails in the learning data are in plain text format. We need to convert the plain text into features that can represent the emails. Using these features we can then use a learning algorithm on the emails. A number of preprocessing steps are first performed.

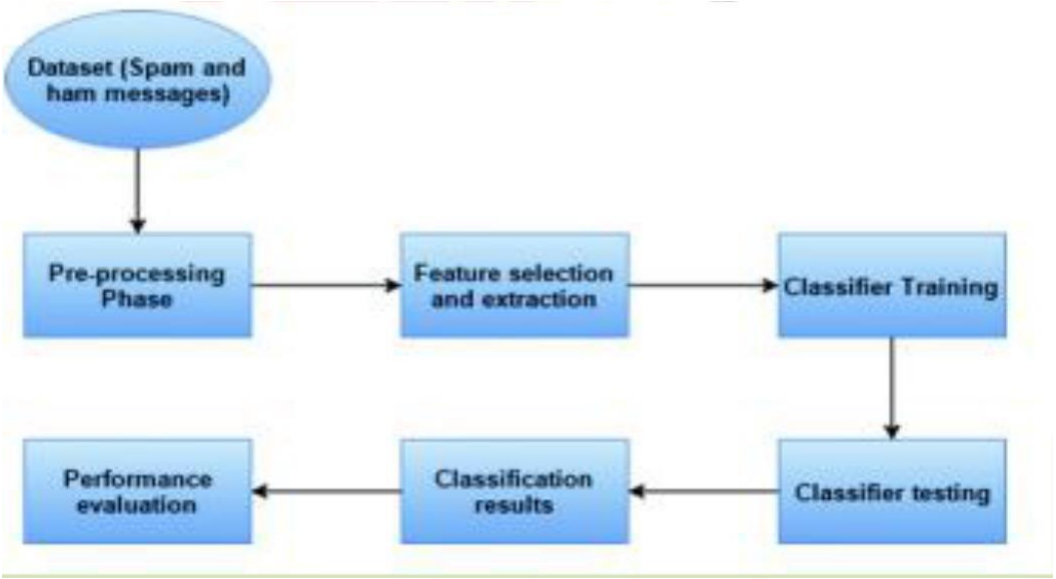
STOP WORDS

There are some English words which appear very frequently in all documents and so have no worth in representing the documents.

These are called STOP WORDS and there is no harm in deleting them. Example: the, a, for etc. There are also some domain specific (in this case email) stop words such as mon, tue, email, sender, from etc. So, we delete these words from all the files using a Bourne Shell Script. These words are put in a file 'words.txt'. The shell script takes multiple files as an argument and then deletes all the stop words mentioned in the words.txt file.

STEMMING

The next step to be performed is stemming. Stemming is used to find a root of a word and thus replacing all words to their stem which reduces the number of words to be considered for representing a document. Example: sings, singing, sing have sing as their stem. In the project, we use PYTHON implementation of Porter stemming algorithm which is slightly modified to meet our needs.



STEP FOR DATA PROCESSING

- LOWER CASE
- TOKENIZATION
- REMOVING SPECIAL CHARACTER

- REMOVING STOP WORD
- PUNCTUATION
- STEMMING

MODEL BUILDING

HERE WE USE THREE MODEL BUILDING ALGORITHM

- GNB = Gaussian NB()
- MNB = Multinomial NB()
- BNB= Bernoulli NB()

VISUALIZATION USING WORDCLOUDS

WordCloud is an approach to outwardly delineate the recurrence at which words show up in information. The cloud is comprised of words scattered fairly haphazardly around the figure.

Words seeming all the more regularly in the content are appeared in a bigger text style, while less normal terms are appeared in littler textual styles. This sort of figure has developed in fame as of late since it gives an approach to watch trending activities on social networking sites.

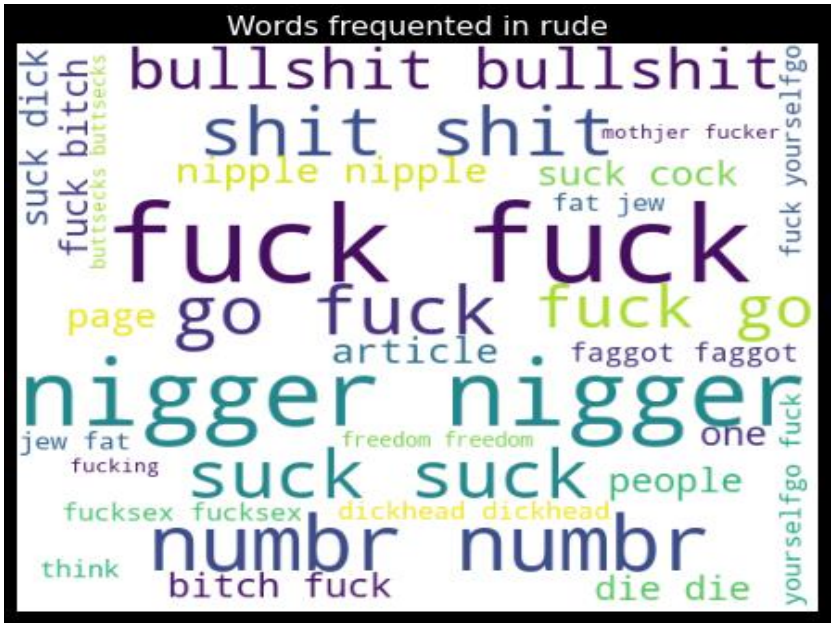
Word Clouds:

OBSERVATIONS:

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

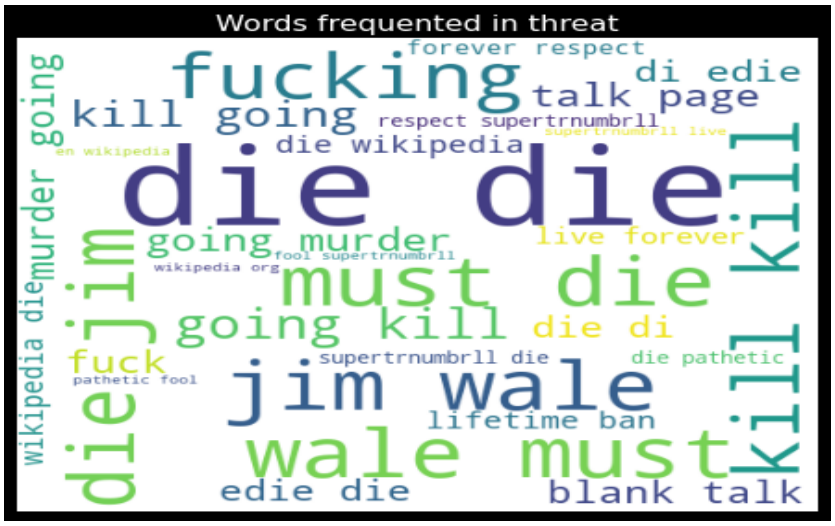
OBSERVATIONS:

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



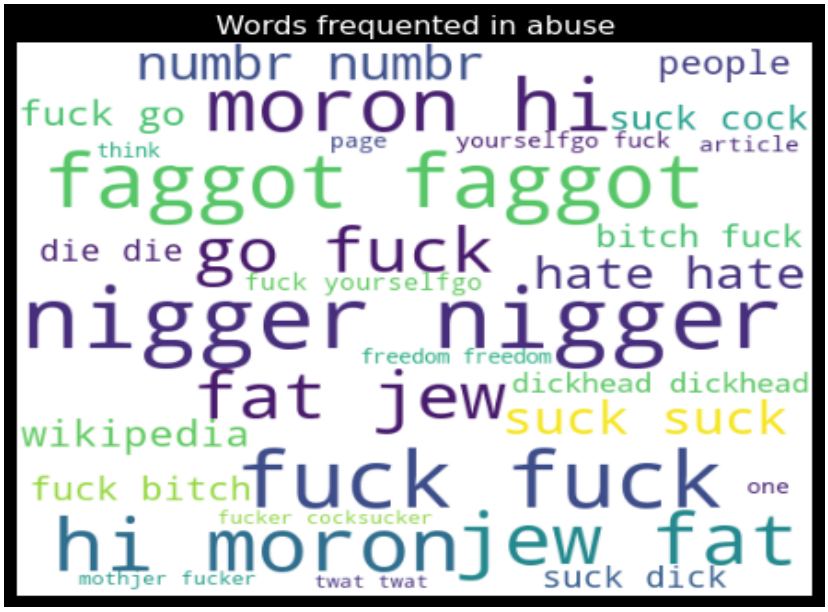
OBSERVATIONS:

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



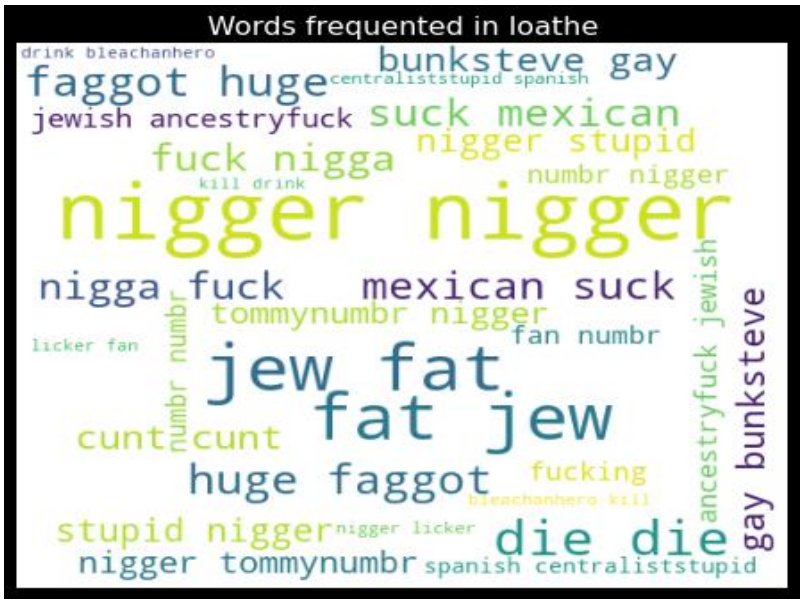
OBSERVATIONS:

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



OBSERVATIONS:

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



OBSERVATIONS:

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.
- 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 labelled 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.

We compare the word clouds of ham and spam messages and See the difference between the frequently occurring terms in in the dataset

RESULT OF MODEL BUILDING

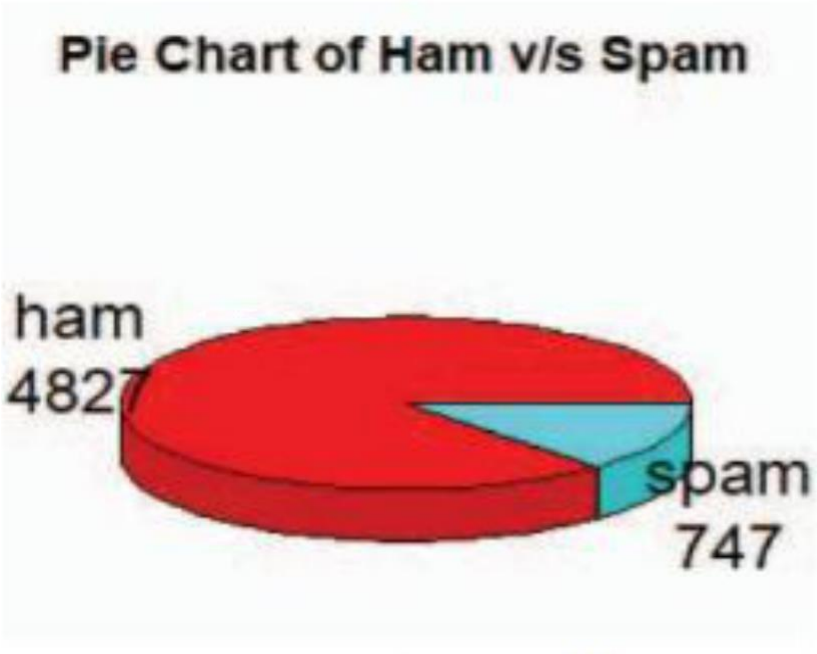
1. GNB = Gaussian NB()
ACCURACY SCORE-88%
PRECISION SCORE-53%
2. MNB = Multinomial NB()
ACCURACY SCORE- 96%
PRECISION SCORE -83%
3. BNB= Bernoulli NB()
ACCURACY SCORE- 97%
PRECISION SCORE -97%

Cross Validation 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 CLASSIFIER 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 %.

```
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'}
```

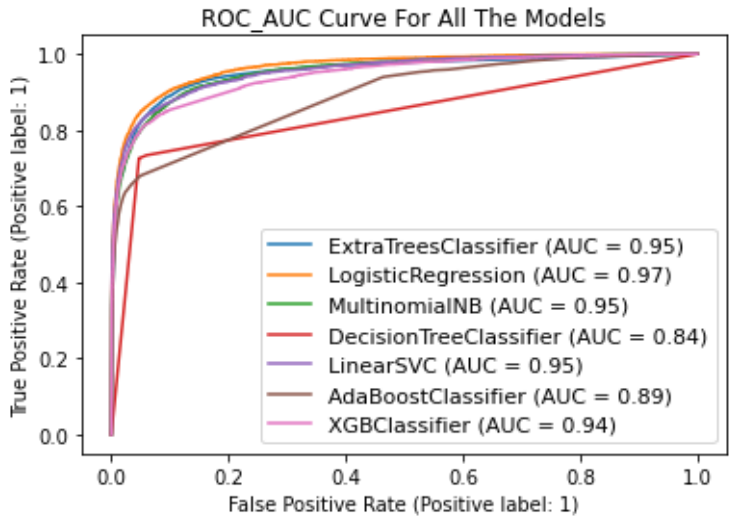
HYPER PARAMETER TUNING [FINAL MODEL]:

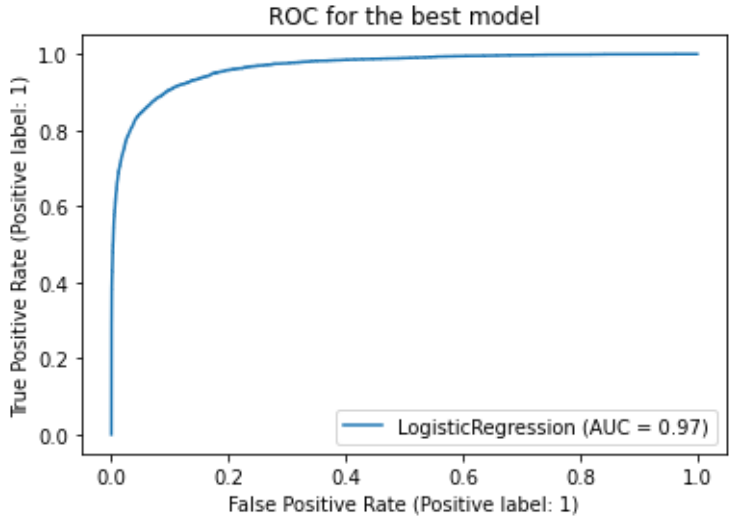
```
In [124]: Final_Model= LogisticRegression(C=0.4,penalty='l2',solver='newton-cg',multi_class='auto')

          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
```

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.

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:

The aims and objectives of the project, which achieved throughout the course, defined at the very first stage of the process. To collect all the information, the research work involved a careful study on the

different filtering algorithms and existing anti-spam tools. These large scale research papers and existing software programs are one of the sources of inspiration behind this project work. The whole project was divided into several iterations. Each iteration was completed by completing four phases: inception, where the idea of work was identified; elaboration, where architecture of the part of the system is designed; construction, where existing code is implemented; transition, where the developed part of the project is validated. However, there are still some parts that can be improved: for example, adding additional filtering techniques or changing aspects of the existing ones. The changes such as incrementing or decrementing the number of interesting words of the message and reorganizing the formula for calculating interesting rate can be done later.

FUTURE SCOPE

In the future, we plan to deal with more challenging problems such as the analysis and management of report in spam SMS filters storing. Solution for this problem is another focus of work in the future.