Malignant Comment Classifier PROJECT

Submitted by:

AKSHAY DINESH SHAH

ACKNOWLEDGMENT

I express my sincere gratitude to Flip Robo Technologies for giving me the opportunity to work on this project on Malignant Comment Classifier using machine learning algorithms and NLTK suite of libraries and also, for providing me with the requisite datasets for training and testing prediction accuracies of the models.

INTRODUCTION

- ## Business Problem Framing

  With the proliferation of social media there has been an emergence of conflict and hate, making online environments uninviting for users. There is a lack of models for online hate detection. Online hate, described as abusive language, aggression, cyberbullying, hatefulness and many others has been identified as a major threat on online social media platforms. Social media platforms are the most prominent grounds for such toxic behaviour. Our goal is to build a prototype of online hate and abuse comment classifier which can used to classify hate and offensive comments so that it can be controlled and restricted from spreading hatred and cyberbullying.

- ## Conceptual Background of the Domain Problem

  Predictive modelling, Classification algorithms are some of the machine learning techniques used along with the various libraries of the NLTK suite for Classification of comments.

  Using NLTK tools, the frequencies of malignant words occurring in textual data were estimated and given appropriate weightage, whilst filtering out words, and other noise which do not have any impact on the semantics of the comments and reducing the words to their base lemmas for efficient processing and accurate classification of the comments.

- ## Review of Literature

  Two research papers titled: “Toxic Comment Classification” by Sara Zahiri and “Machine learning methods for toxic comment classification: a systematic review” by Darko Androcec were reviewed and studied to gain insights into the nature of malignant comments, their impact on social media platforms and the various

methods that are employed for training models to detect, identify and classify them.

- Motivation for the Problem Undertaken

  There has been a remarkable increase in the cases of cyberbullying and trolls on various social media platforms. Many celebrities and influences are facing backlashes from people and have to come across hateful and offensive comments. This can take a toll on anyone and affect them mentally leading to depression, mental illness, self-hatred and suicidal thoughts.

  Internet comments are bastions of hatred and vitriol. While online anonymity has provided a new outlet for aggression and hate speech, machine learning can be used to fight it. The problem we sought to solve was the tagging of internet comments that are aggressive towards other users. This means that insults to third parties such as celebrities will be tagged as unoffensive, but “u are an idiot” is clearly offensive. Automatic recognition of malignant comments on online forums, and social media serves as a useful provision for moderators of public platforms as well as users who could receive warnings and filter unwanted contents. The need of advanced methods and techniques to improve identification of different types of comments posted online motivated the current project.

Analytical Problem Framing

- Mathematical/ Analytical Modeling of the Problem

  Various Classification analysis techniques were used to build Classification models to determine whether an input Message content is benign or malignant. Machine Learning Algorithms such as Multinomial Naïve Bayes and Complement Naïve Bayes were employed which are based on the Bayes Theorem:

  P(message is malignant | message content) = P(message content | malignant). P(malignant) / P(message content)

  The probability of message being Malignant, knowing that Message Content has occurred could be calculated. Event of “Message Content” represents the evidence and “Message is Malignant”, the hypothesis to be approved. The theorem runs on the assumption that all predictors/features are independent and the presence of one would not affect the other.

  The approach to classify a comment as malignant would depend on training data labelled as various categories of malignant messages and benign messages.

- Data Sources and their formats

  - The data set contains the training set, which has 159571 samples and the test set which contains 153164 samples. All the data samples contain 8 fields which includes ‘Id’, ‘Comments’, ‘Malignant’, ‘Highly malignant’, ‘Rude’, ‘Threat’, ‘Abuse’ and ‘Loathe’.
  - The label can be either 0 or 1, where 0 denotes a NO while 1 denotes a YES. There are various comments which have multiple labels. The first attribute is a unique ID associated with each comment.

The data set includes:

- **Malignant:** It is the Label column, which includes values 0 and 1, denoting if the comment is malignant or not.
- **Highly Malignant:** It denotes comments that are highly malignant and hurtful.
- **Rude:** It denotes comments that are very rude and offensive.
- **Threat:** It contains indication of the comments that are giving any threat to someone.
- **Abuse:** It is for comments that are abusive in nature.
- **Loathe:** It describes the comments which are hateful and loathing in nature.
- **ID:** It includes unique Ids associated with each comment text given.
- **Comment text:** This column contains the comments extracted from various social media platforms.

- Data Preprocessing Done

  The dataset was checked to see if there were any null values or random characters present. None were found.

  Column: **ID** was dropped since they don't contribute to building a good model for predicting the target variable values.

```
# Convert all messages to lower case
df['comment_text'] = df['comment_text'].str.lower()

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

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

# Replace money symbols with 'moneysymb' \
df['comment_text'] = df['comment_text'].str.replace(r'£|\$', 'dollars')

# Replacing 10 digit phone numbers with 'phonenumber'
df['comment_text'] = df['comment_text'].str.replace(r'^\(?[\d]{3}\)?[\s-]?[\d]{3}[\s-]?[\d]{4}$','phonenumber')

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

#removing punctuations
df['comment_text'] = df['comment_text'].str.replace(r'[^\w\d\s]',' ')

#removing underscore characters
df['comment_text'] = df['comment_text'].str.replace(r'[\_]',' ')

#removing single characters
df['comment_text'] = df['comment_text'].str.replace(r'\s+[a-zA-Z]\s+', ' ')

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

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

The train and test dataset contents were then converted into lowercase. Punctuations, unnecessary characters etc were removed, currency symbols, phone numbers, web urls, email addresses etc. were replaced with single words. Tokens that contributed nothing to semantics of the messages were remove as Stop words. Finally retained tokens were lemmatized using WordNetLemmatizer().

The string lengths of original comments and the cleaned comments were then compared.

- State the set of assumptions (if any) related to the problem under consideration

  The comment content made available in Train and Test Dataset is assumed to be written in English Language in the standard Greco-Roman script. This is so that the Stop word package and WordNetLemmatizer can be effectively used.

- Hardware and Software Requirements and Tools Used

Hardware Used:

- Processor: Intel i2 10th generation
- Physical Memory: 8.0GB (3200MHz)

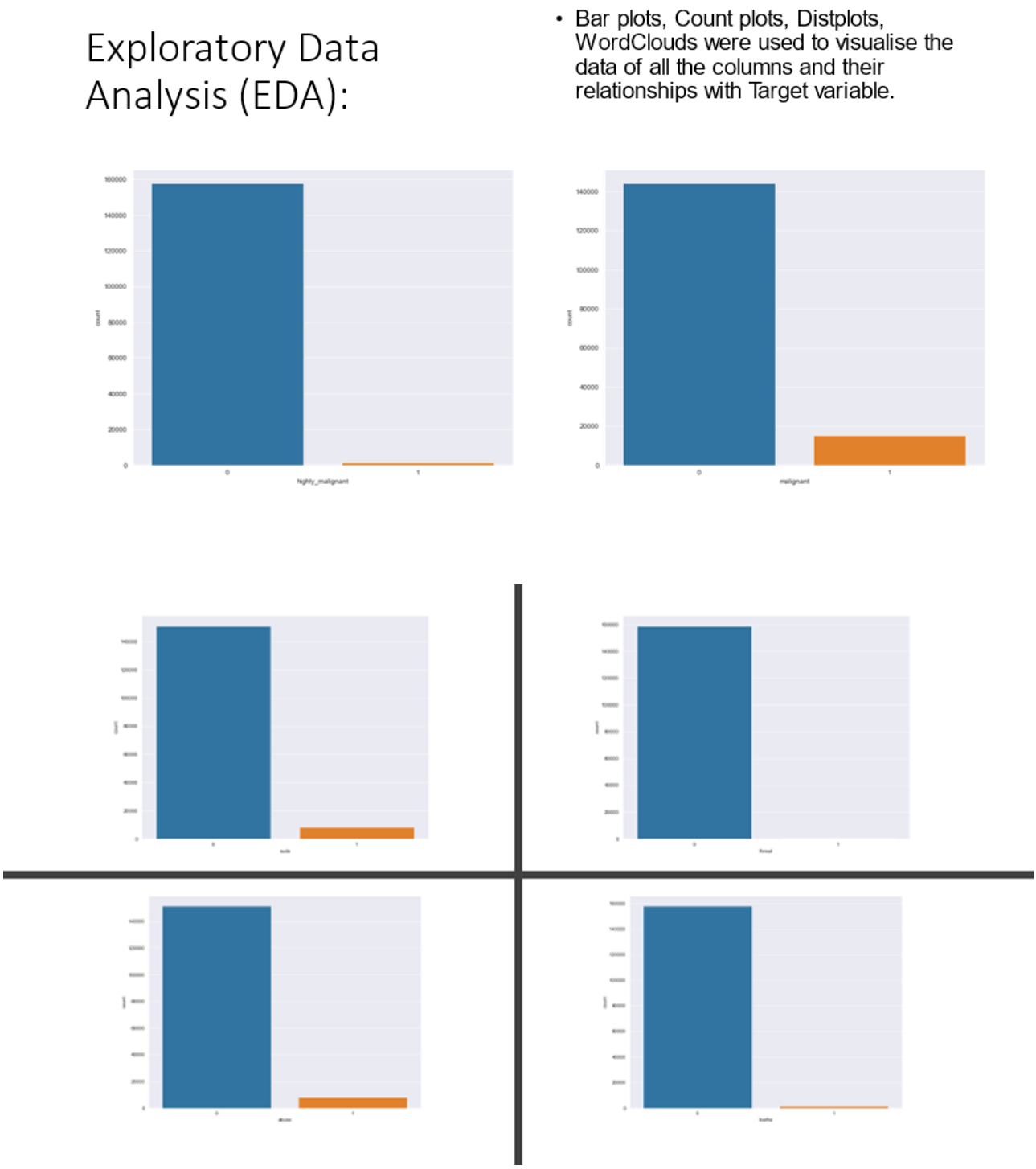
Software Used:

- Windows 11 Operating System
- Anaconda Package and Environment Manager: Anaconda is a distribution of the Python programming language for scientific computing, that aims to simplify package management and deployment.The distribution includes data science packages suitable for Windows and provides a host of tools and environment for conducting Data Analytical and Scientific works. Anaconda provides all the necessary Python packages and libraries for Machine learning projects.

Data Visualization

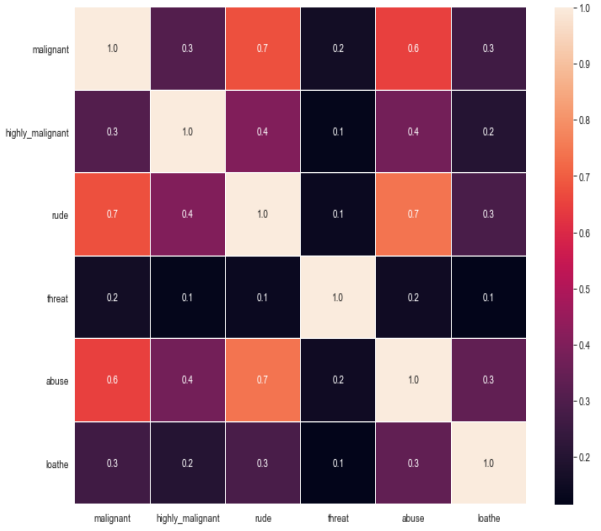
Exploratory Data Analysis

Barplots, Countplots, Distplots, WordClouds were used to visualize the data of all the columns and their relationships with Target variable.



Correlation

Correlation:




```
df.corr()
```

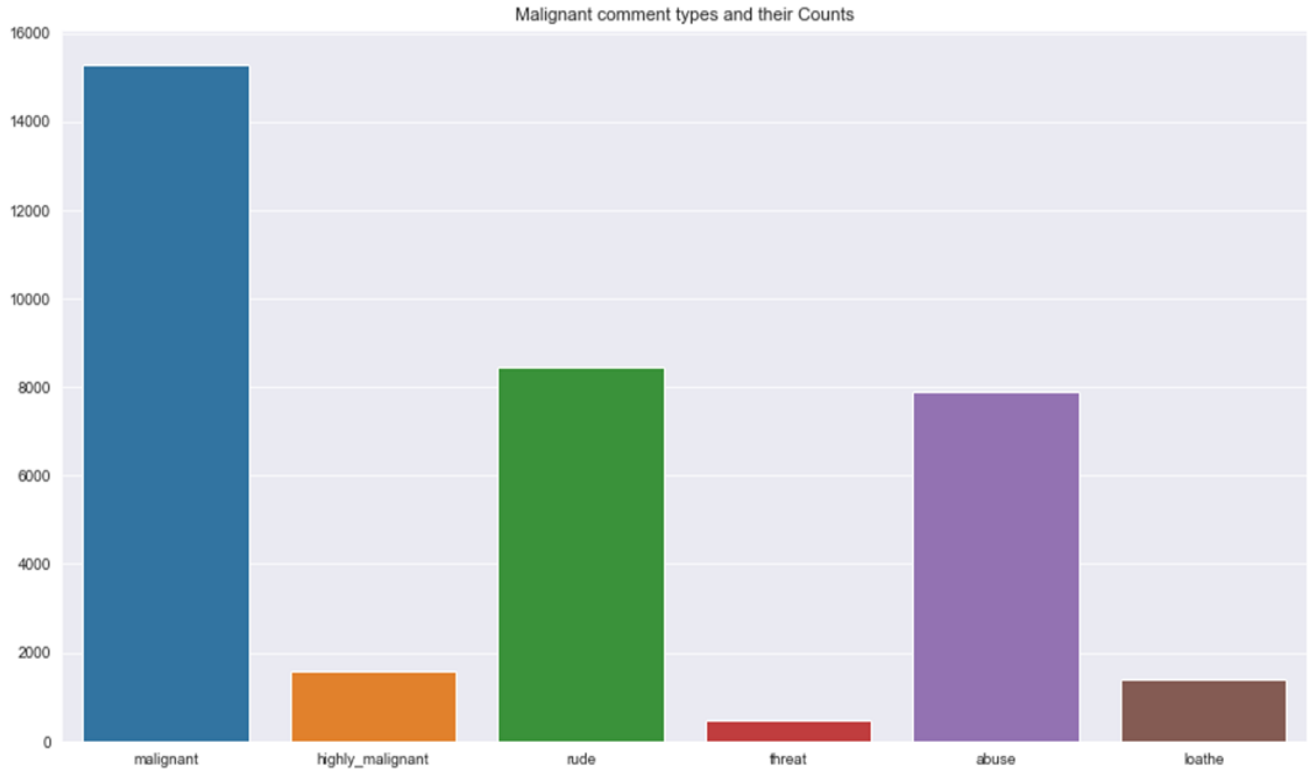
	malignant	highly_malignant	rude	threat	abuse	loathe
malignant	1.000000	0.308619	0.676515	0.157058	0.647518	0.266009
highly_malignant	0.308619	1.000000	0.403014	0.123601	0.375807	0.201600
rude	0.676515	0.403014	1.000000	0.141179	0.741272	0.286867
threat	0.157058	0.123601	0.141179	1.000000	0.150022	0.115128
abuse	0.647518	0.375807	0.741272	0.150022	1.000000	0.337736
loathe	0.266009	0.201600	0.286867	0.115128	0.337736	1.000000

From the graphs above it is observed that columns: Rude,Abuse, Malignant have highest positive correlation with comment_type.

Malignant


Malignant Comment
suck suck
make
suck cock
fuck yourselfgo
moron hi
hate hate
one
fucking
think
know
die die
suck dick
fucksex fucksex
want
nipple nipple
shit shit
fuck fuck
jew fat
say
edit
way
life
thing
stupid
article
people
wikipedia
talk page
faggot faggot
even
bark bark
pig pig
num num
see
really
page
go fuck
bitch fuck
fat jew
time
die
fag
bullshit bullshit
fuck bitch
nigger nigger
dickhead dickhead
die fag
hi moron
wanker wanker
ball ball
fuck go

Above graphs show that the string length of comments was drastically brought down after processing.



The above graph shows the composition of toxic comments, of which majority are malignant followed by rude comments, abusivecomments, highly malignant comments, hateful comments and threats.

Highly_Malignant

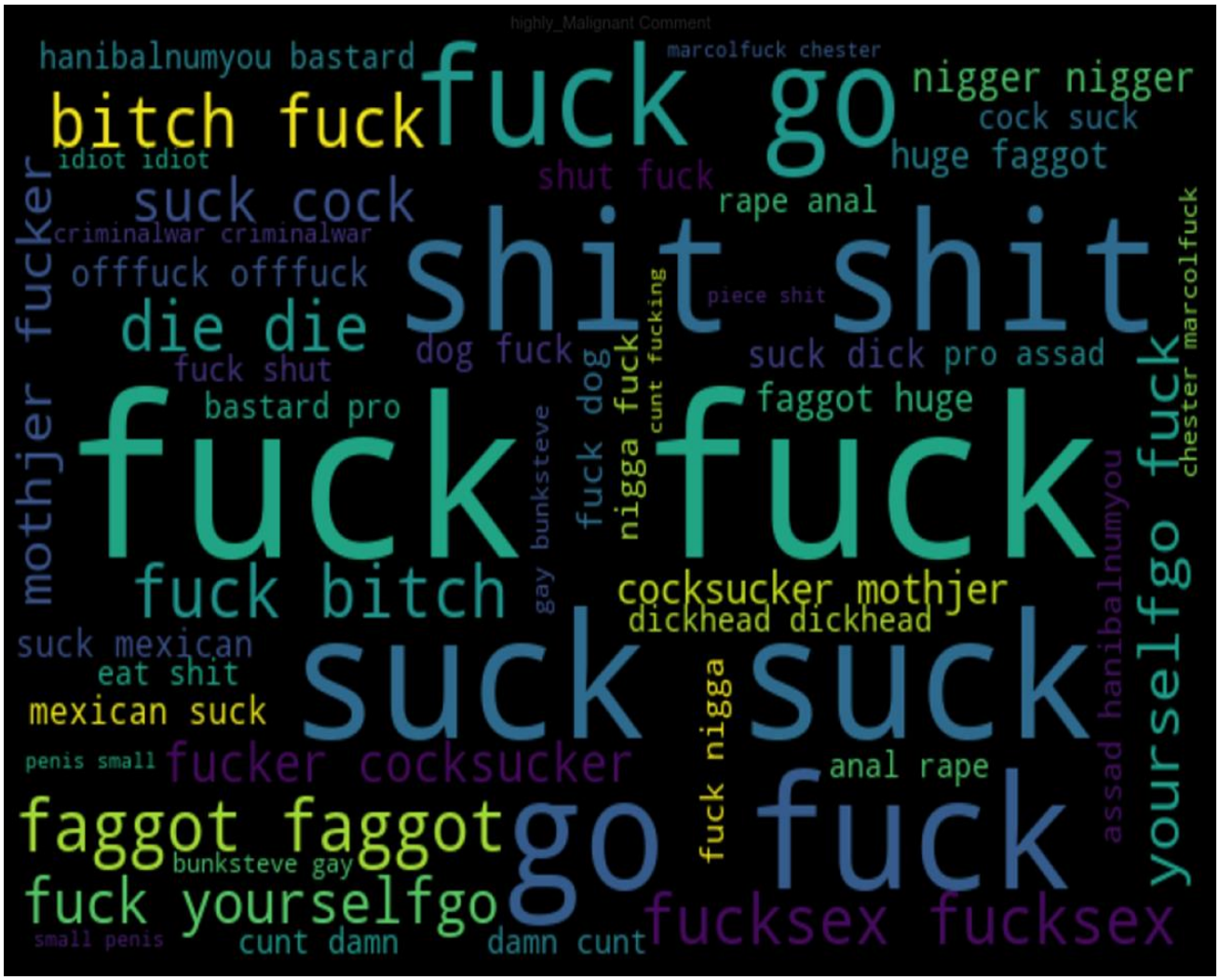
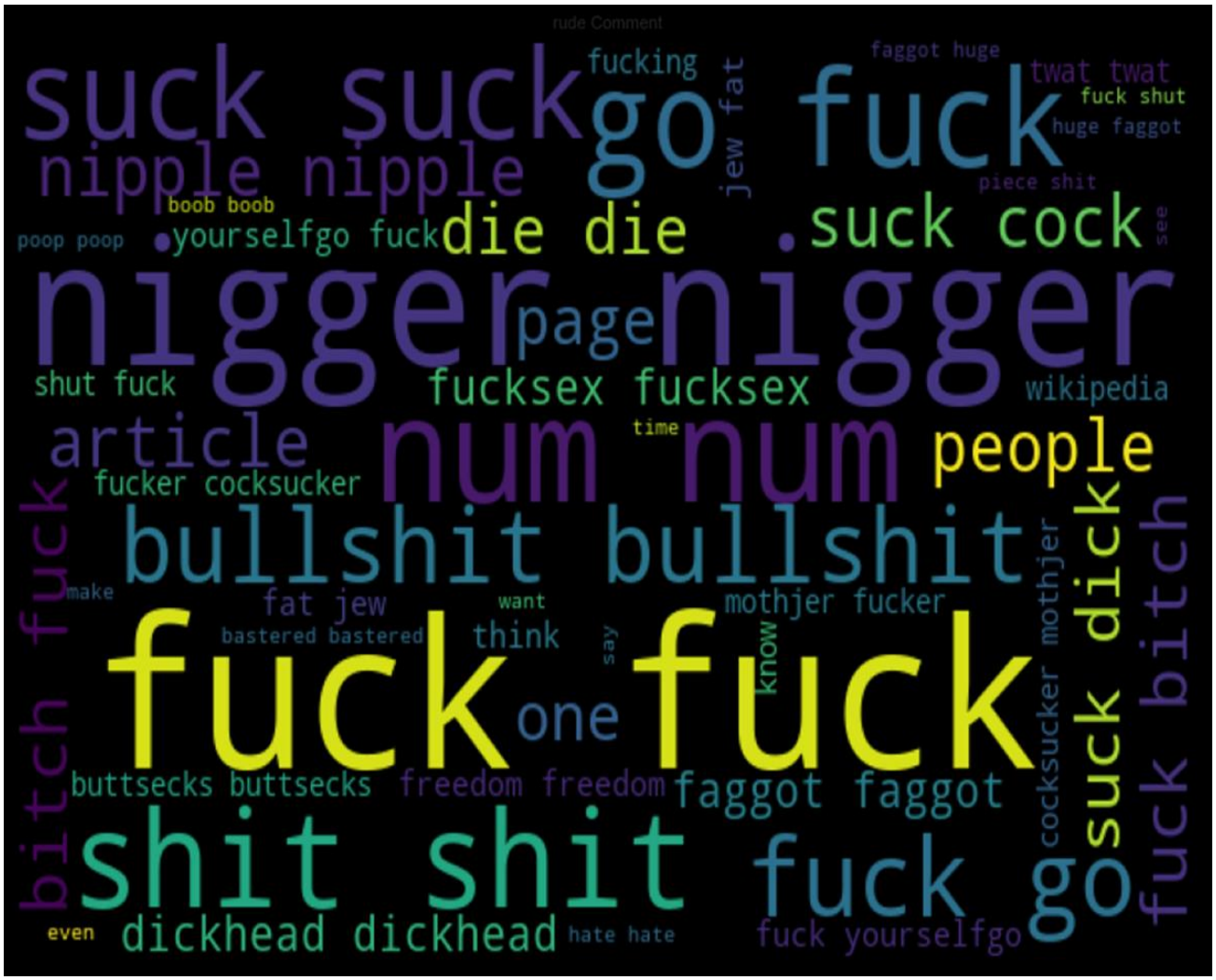


Figure 5 Malignant Words

Rude


rude Comment
suck suck
go
fuck
fucking
jew fat
faggot huge
twat twat
fuck shut
huge faggot
nipple nipple
boob boob
piece shit
poop poop
yourselfgo fuck
die die
.suck cock
see
nigger
nigger
page
shut fuck
fucksex fucksex
wikipedia
article
num num
time
people
fucker cocksucker
bullshit bullshit
bitch fuck
make
fat jew
want
mothjer fucker
suck dick
fuck bitch
bastered bastered
think
say
know
fuck
fuck
one
cocksucker mothjer
buttsecks buttsecks
freedom freedom
faggot faggot
shit shit
fuck
go
even
dickhead dickhead
hate hate
fuck yourselfgo

Threat

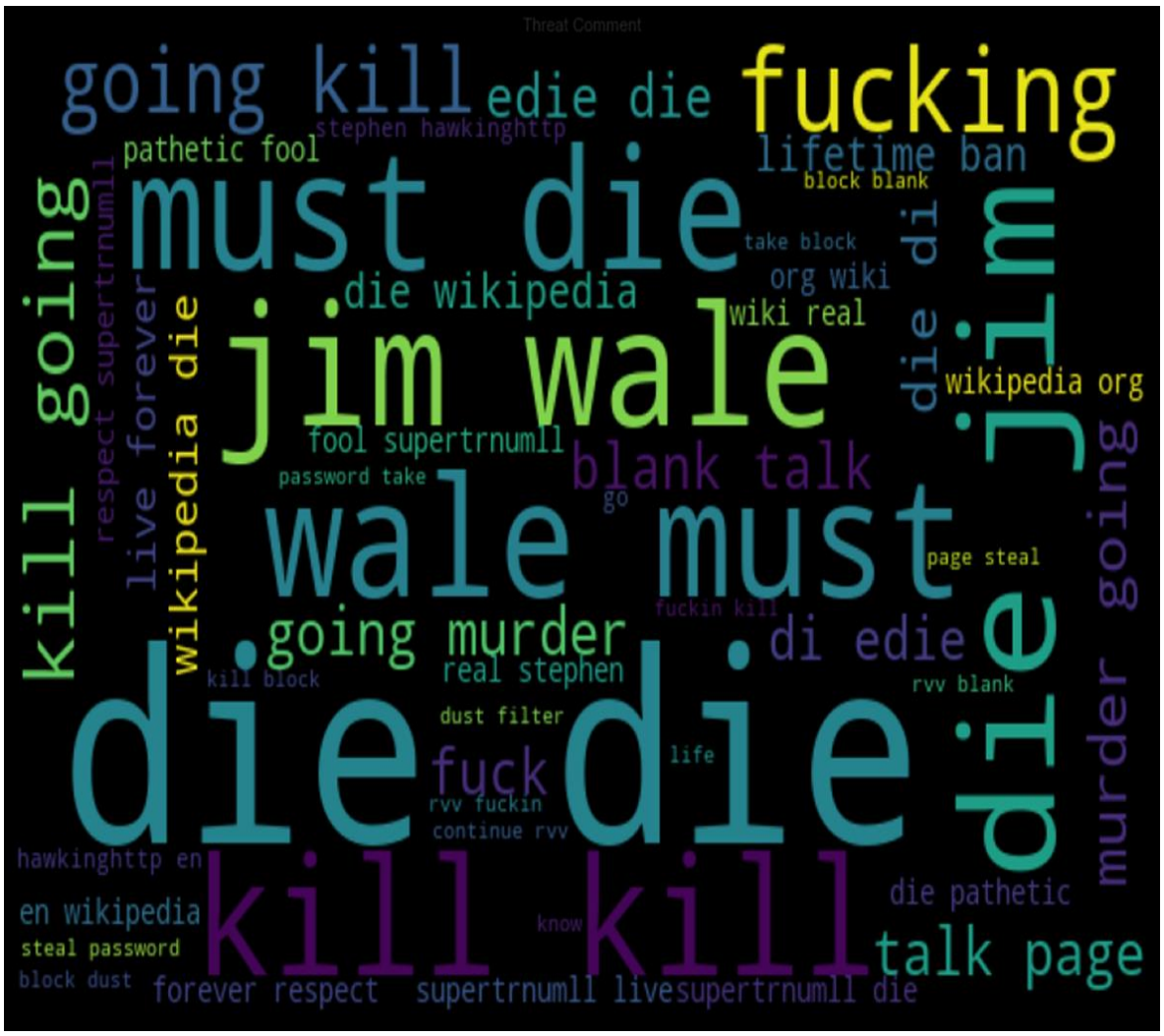


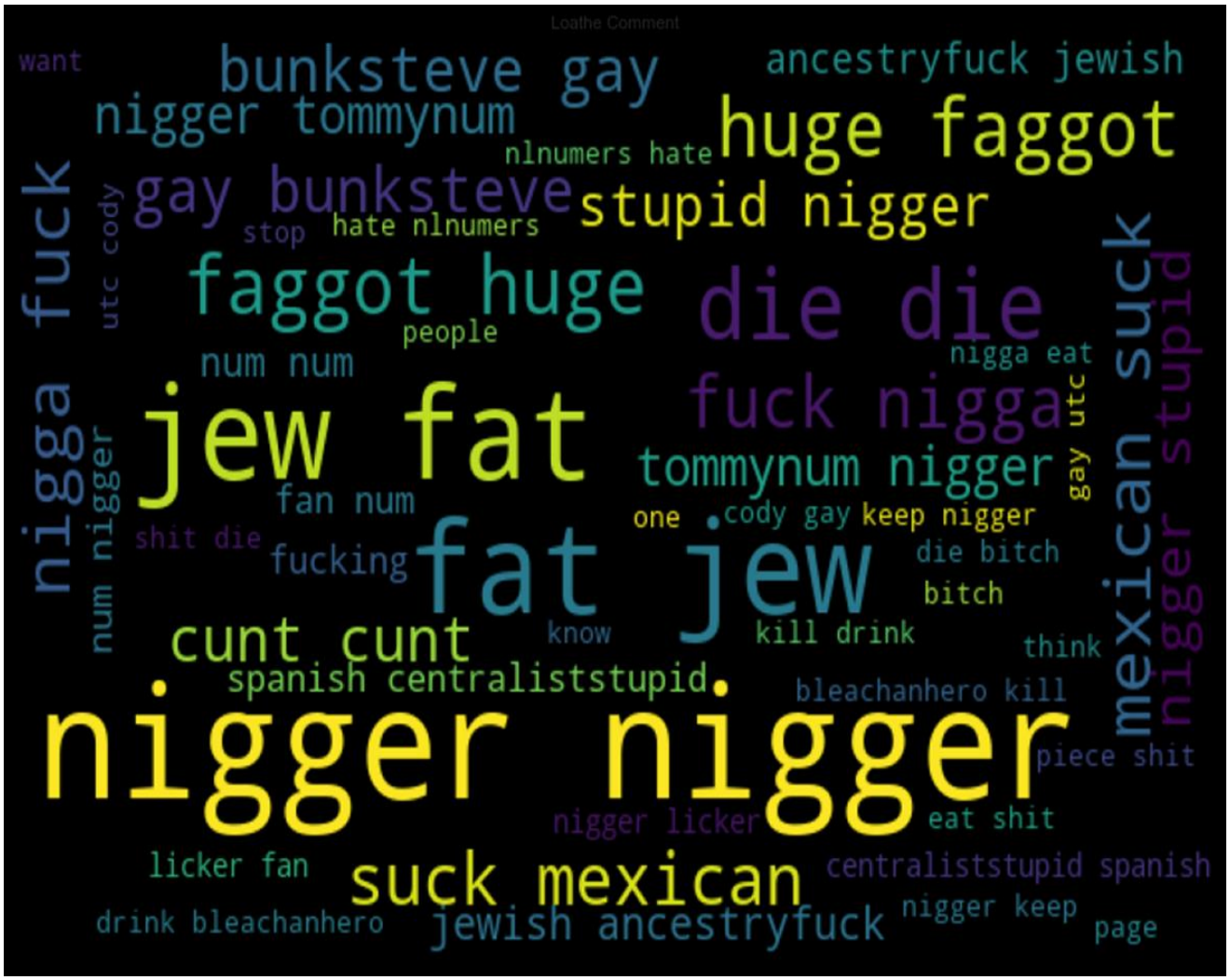
Figure 7 Highly Malignant Words

Figure 8 Threat Words

Abusive

Figure 9 Abusive Words

Loathe



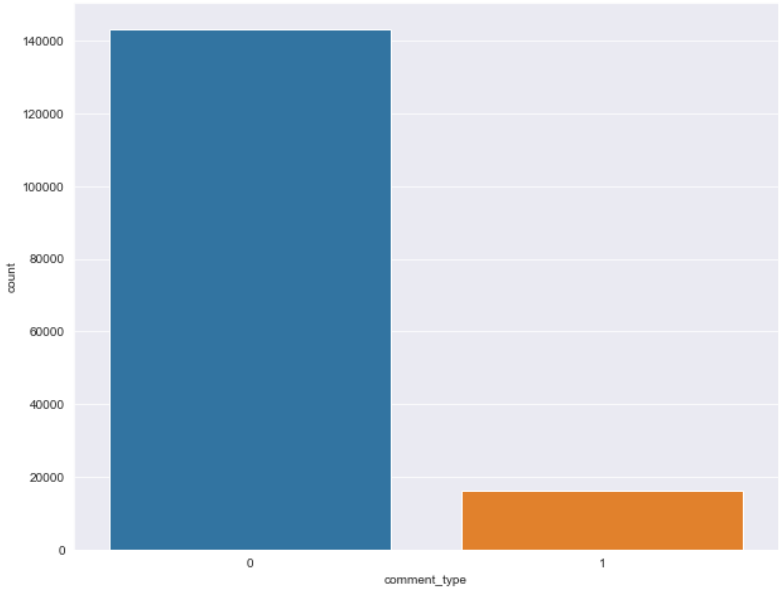
Feature Engineering

The comments data could belong to more than one label simultaneously (rude comments are at the same time malignant and in some cases can also be deemed hateful, abusive comments are hateful and can be highly malignant at the same time, threats are highly malignant too etc.)

Since each of the categories had very small data available to work with, a new column: ‘comment_type’ was created which only had binary classes: 0 which represented all the benign comments and 1 which represented all the comments which fell under malignant, highly malignant, abusive, hateful, rude, threat features. This column acted as Target Label column for malignant comment classification.

Visualizing data in Target column

Visualising data in Target column



The classes appear to be imbalanced with 90% of comments being benign (0) and only 10% being malignant (1).

Model/s Development and Evaluation

- Identification of possible problem-solving approaches (methods)
  The model algorithms used were as follows:

- Logistic Regression: It is a classification algorithm used to find the probability of event success and event failure. It is used when the dependent variable is binary (0/1, True/False, Yes/No) in nature. It supports categorizing data into discrete classes by studying the relationship from a given set of labelled data. It learns a linear relationship from the given dataset and then introduces a non-linearity in the form of the Sigmoid function. It not only provides a measure of how appropriate a predictor (coefficient size)is, but alsoits direction of association (positive or negative).
- Multinomial Naïve Bayes Classifier: Multinomial Naive Bayes algorithm is a probabilistic learning method that is mostly used in Natural Language Processing (NLP). The algorithm is based on the Bayes theorem. It calculates the probability of each tag for a

given sample and then gives the tag with the highest probability as output.

- XGBClassifier: XGBoost uses decision trees as base learners; combining many weak learners to make a strong learner. As a result it is referred to as an ensemble learning method since it uses the output of many models in the final prediction. It uses the power of parallel processing and supports regularization.
- RandomForestClassifier: A random forest is a meta estimator that fits a number of classifying decision trees on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. A random forest produces good predictions that can be understood easily. It reduces overfitting and can handle large datasets efficiently. The random forest algorithm provides a higher level of accuracy in predicting outcomes over the decision tree algorithm.
- Complement Naïve Bayes Classifier: Complement Naive Bayes is somewhat an adaptation of the standard Multinomial Naive Bayes algorithm. Complement Naive Bayes is particularly suited to work with imbalanced datasets. In complement Naive Bayes, instead of calculating the probability of an item belonging to a certain class, we calculate the probability of the item belonging to all the classes.
- Passive Aggressive Classifier: Passive-Aggressive algorithms do not require a learning rate and are called so because if the prediction is correct, keep the model and do not make any changes. i.e., the data in the example is not enough to cause any changes in the model. If the prediction is incorrect, make changes to the model. i.e., some change to the model may correct it.
- AdaBoost Classifier: The basis of this algorithm is the Boosting main core: give more weight to the misclassified observations. the meta-learner adapts based upon the results of the weak classifiers, giving more weight to the misclassified observations of the last weak learner. The individual learners can be weak, but as long as the performance of each weak learner is better than random guessing, the final model can converge to a strong learner (a learner not influenced by outliers and with a great generalization power, in order to have strong performances on unknown data).

Best Random state was found to be 56

```
from sklearn.naive_bayes import MultinomialNB
maxAcc = 0
maxRS=0
for i in range(0,100):
    x_train,x_test,y_train,y_test = train_test_split(smt_x,smt_y,test_size = .30, random_state = i)
    modRF =  MultinomialNB()
    modRF.fit(x_train,y_train)
    pred = modRF.predict(x_test)
    acc  = accuracy_score(y_test,pred)
    if acc>maxAcc:
        maxAcc=acc
        maxRS=i
print(f"Best Accuracy is: {maxAcc} on random_state: {maxRS}")
```

```
Best Accuracy is: 0.909566551948656 on random_state: 56
```

Training the Models

```
RFC.fit(x_train,y_train)
XGBC.fit(x_train,y_train)
adbc.fit(x_train,y_train)
LOGR.fit(x_train,y_train)
MNB.fit(x_train,y_train)
CNB.fit(x_train,y_train)
```

```
pc.fit(x_train,y_train)
```

```
PassiveAggressiveClassifier()
```

All Models have been trained.

- **Analyzing Accuracy of The Models**

Classification Report consisting of Precision,Recall, Support and F1-score were the metrics used to evaluate the Model Performance.

Precision is defined as the ratio of true positives to the sum of true and false positives. Recall is defined as the ratio of true positives to the sum of true positives and false negatives.The F1 is the weighted harmonic mean of precision and recall. The closer the value of the F1 score is to 1.0, the better the expected performance of the model is. Support is the number of actual occurrences of the class in the dataset. It doesn’t vary between models; it just diagnoses the performance evaluation process.

Log Loss quantifies the accuracy of a classifier by penalizing false classifications.

LogisticRegression

```
lr = LogisticRegression(solver='liblinear')
lr.fit(x_train,y_train)
```

```
LogisticRegression(solver='liblinear')
```

```
lr.score(x_train,y_train)
```

```
0.9593013366279017
```

```
pred_lr = lr.predict(x_test)
```

```
print('Classification Report:',classification_report(y_test, pred_lr))
print('Confusion Matrix:',confusion_matrix(y_test,pred_lr))
print('Log_Loss:',log_loss(y_test,pred_lr))
```

```
Classification Report:              precision    recall  f1-score   support

           0       0.96      1.00      0.98     43057
           1       0.93      0.60      0.73      4815

    accuracy                           0.96     47872
   macro avg       0.95      0.80      0.85     47872
weighted avg       0.95      0.96      0.95     47872

Confusion Matrix: [[42854   203]
 [ 1935  2880]]
Log_Loss: 1.5425314641251127
```

RandomForestClassifier

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

```
RandomForestClassifier()
```

```
RFC.score(x_train,y_train)
```

```
0.9994717947340621
```

```
pred_rfc = RFC.predict(x_test)
```

```
print('Classification Report:',classification_report(y_test, pred_rfc))
print('Confusion Matrix:',confusion_matrix(y_test,pred_rfc))
print('Log_Loss:',log_loss(y_test,pred_rfc))
```

```
Classification Report:              precision    recall  f1-score   support

           0       0.96      0.99      0.98     43057
           1       0.90      0.63      0.74      4815

    accuracy                           0.96     47872
   macro avg       0.93      0.81      0.86     47872
weighted avg       0.95      0.96      0.95     47872

Confusion Matrix: [[42736   321]
 [ 1783  3032]]
Log_Loss: 1.518003054095655
```

AdaBoostClassifier

```
ad = AdaBoostClassifier()
ad.fit(x_train,y_train)
```

```
AdaBoostClassifier()
```

```
ad.score(x_train,y_train)
```

```
0.9463110681384793
```

```
pred_ad = ad.predict(x_test)
```

```
print('Classification Report:',classification_report(y_test, pred_ad))
print('Confusion Matrix:',confusion_matrix(y_test,pred_ad))
print('Log_Loss:',log_loss(y_test,pred_ad))
```

```
Classification Report:              precision    recall  f1-score   support

           0       0.95      0.99      0.97     43057
           1       0.88      0.55      0.68      4815

    accuracy                           0.95     47872
   macro avg       0.91      0.77      0.82     47872
weighted avg       0.94      0.95      0.94     47872

Confusion Matrix: [[42683   374]
 [ 2166  2649]]
Log_Loss: 1.8325700010969266
```

MultinomialNB

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

```
MultinomialNB()
```

```
mt.score(x_train,y_train)
```

```
0.9379940733578636
```

```
pred_mt = mt.predict(x_test)
```

```
print('Classification Report:',classification_report(y_test, pred_mt))
print('Confusion Matrix:',confusion_matrix(y_test,pred_mt))
print('Log_Loss:',log_loss(y_test,pred_mt))
```

```
Classification Report:              precision    recall  f1-score   support

           0       0.93      1.00      0.96     43057
           1       0.97      0.36      0.52      4815

    accuracy                           0.93     47872
   macro avg       0.95      0.68      0.74     47872
weighted avg       0.94      0.93      0.92     47872

Confusion Matrix: [[43011    46]
 [ 3086  1729]]
Log_Loss: 2.259681744032881
```

ComplementNB

```
cp = ComplementNB()
cp.fit(x_train,y_train)
```

```
ComplementNB()
```

```
cp.score(x_train,y_train)
```

```
0.9336520470192213
```

```
pred_cp = cp.predict(x_test)
```

```
print('Classification Report:',classification_report(y_test, pred_cp))
print('Confusion Matrix:',confusion_matrix(y_test,pred_cp))
print('Log_Loss:',log_loss(y_test,pred_cp))
```

```
Classification Report:              precision    recall  f1-score   support

           0       0.97      0.94      0.95     43057
           1       0.57      0.72      0.64      4815

    accuracy                           0.92     47872
   macro avg       0.77      0.83      0.80     47872
weighted avg       0.93      0.92      0.92     47872

Confusion Matrix: [[40488  2569]
 [ 1346  3469]]
Log_Loss: 2.824644129175165
```

PassiveAggressiveClassifie

```
pac = PassiveAggressiveClassifier()
pac.fit(x_train,y_train)
```

```
PassiveAggressiveClassifier()
```

```
pac.score(x_train,y_train)
```

```
0.9933213368069544
```

```
pred_pac = pac.predict(x_test)
```

```
print('Classification Report:',classification_report(y_test, pred_pac))
print('Confusion Matrix:',confusion_matrix(y_test,pred_pac))
print('Log_Loss:',log_loss(y_test,pred_pac))
```

```
Classification Report:              precision    recall  f1-score   support

           0       0.97      0.97      0.97     43057
           1       0.75      0.73      0.74      4815

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

Confusion Matrix: [[41868  1189]
 [ 1321  3494]]
Log_Loss: 1.810939160105496 7
```

XGBClassifier

```
xgb = XGBClassifier()
xgb.fit(x_train,y_train)
```

```
[20:27:15] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.5.1/src/learner.cc:1115: Starting in XGBoost 1.3.
0, the default evaluation metric used with the objective 'binary:logistic' was changed from 'error' to 'logloss'. Explicitly se
t eval_metric if you'd like to restore the old behavior.

XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, enable_categorical=False,
              gamma=0, gpu_id=-1, importance_type=None,
              interaction_constraints='', learning_rate=0.300000012,
              max_delta_step=0, max_depth=6, min_child_weight=1, missing=nan,
              monotone_constraints='()', n_estimators=100, n_jobs=8,
              num_parallel_tree=1, predictor='auto', random_state=0,
              reg_alpha=0, reg_lambda=1, scale_pos_weight=1, subsample=1,
              tree_method='exact', validate_parameters=1, verbosity=None)
```

```
xgb.score(x_train,y_train)
```

```
0.9614678734814098
```

```
pred_xgb = xgb.predict(x_test)
```

```
print('Classification Report:',classification_report(y_test, pred_xgb))
print('Confusion Matrix:',confusion_matrix(y_test,pred_xgb))
print('Log_Loss:',log_loss(y_test,pred_xgb))
```

```
Classification Report:              precision    recall  f1-score   support

           0       0.96      0.99      0.97     43057
           1       0.91      0.60      0.72      4815

    accuracy                           0.95     47872
   macro avg       0.93      0.79      0.85     47872
weighted avg       0.95      0.95      0.95     47872

Confusion Matrix: [[42770   287]
 [ 1943  2872]]
Log_Loss: 1.6089091921188443
```

Model Cross Validation

Cross Validation

AdaBoostClassifier

```
print(cvs(ad,X,y,cv=5).mean())
```

```
0.9461618981083729
```

MultinomialNB

```
print(cvs(mt,X,y,cv=5).mean())
```

```
0.9354644618064345
```

ComplementNB

```
print(cvs(cp,X,y,cv=5).mean())
```

```
0.9138878535295945
```

PassiveAggressiveClassifie

```
print(cvs(pac,X,y,cv=5).mean())
```

```
0.9476784578042758
```

RandomForestClassifier

```
print(cvs(RFC,X,y,cv=5).mean())
```

```
0.9560446427723747
```

LogisticRegression

```
print(cvs(lr,X,y,cv=5).mean())
```

```
0.9550795534030136
```

XGBClassifier

```
print(cvs(xgb,X,y,cv=5).mean())
```

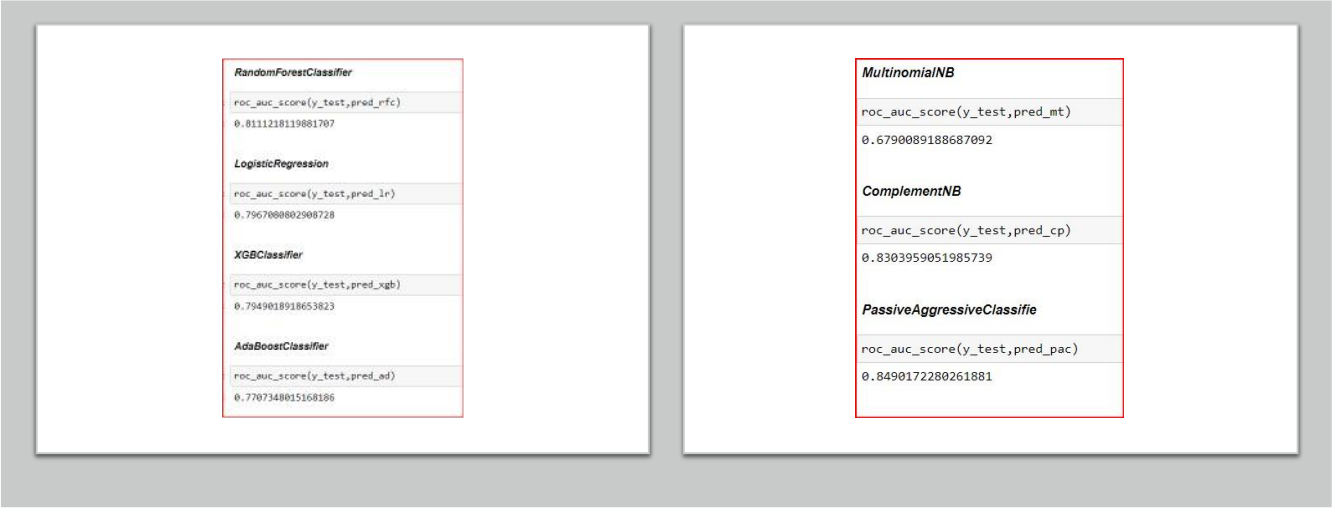
```
[21:11:46] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.5.1/src/learner.cc:1115: Starting in XGBoost 1.3.
0, the default evaluation metric used with the objective 'binary:logistic' was changed from 'error' to 'logloss'. Explicitly se
t eval_metric if you'd like to restore the old behavior.
[21:12:14] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.5.1/src/learner.cc:1115: Starting in XGBoost 1.3.
0, the default evaluation metric used with the objective 'binary:logistic' was changed from 'error' to 'logloss'. Explicitly se
t eval_metric if you'd like to restore the old behavior.
[21:12:41] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.5.1/src/learner.cc:1115: Starting in XGBoost 1.3.
0, the default evaluation metric used with the objective 'binary:logistic' was changed from 'error' to 'logloss'. Explicitly se
t eval_metric if you'd like to restore the old behavior.
[21:13:10] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.5.1/src/learner.cc:1115: Starting in XGBoost 1.3.
0, the default evaluation metric used with the objective 'binary:logistic' was changed from 'error' to 'logloss'. Explicitly se
t eval_metric if you'd like to restore the old behavior.
[21:13:38] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.5.1/src/learner.cc:1115: Starting in XGBoost 1.3.
0, the default evaluation metric used with the objective 'binary:logistic' was changed from 'error' to 'logloss'. Explicitly se
t eval_metric if you'd like to restore the old behavior.
0.9535316530165261
```

ROC AUC Scores

The score is used to summarize the trade-off between the true positive rate and false positive rate for a predictive model using different probability thresholds. The AUC value lies between 0.5 to 1 where 0.5 denotes a bad classifier and 1 denotes an excellent classifier.

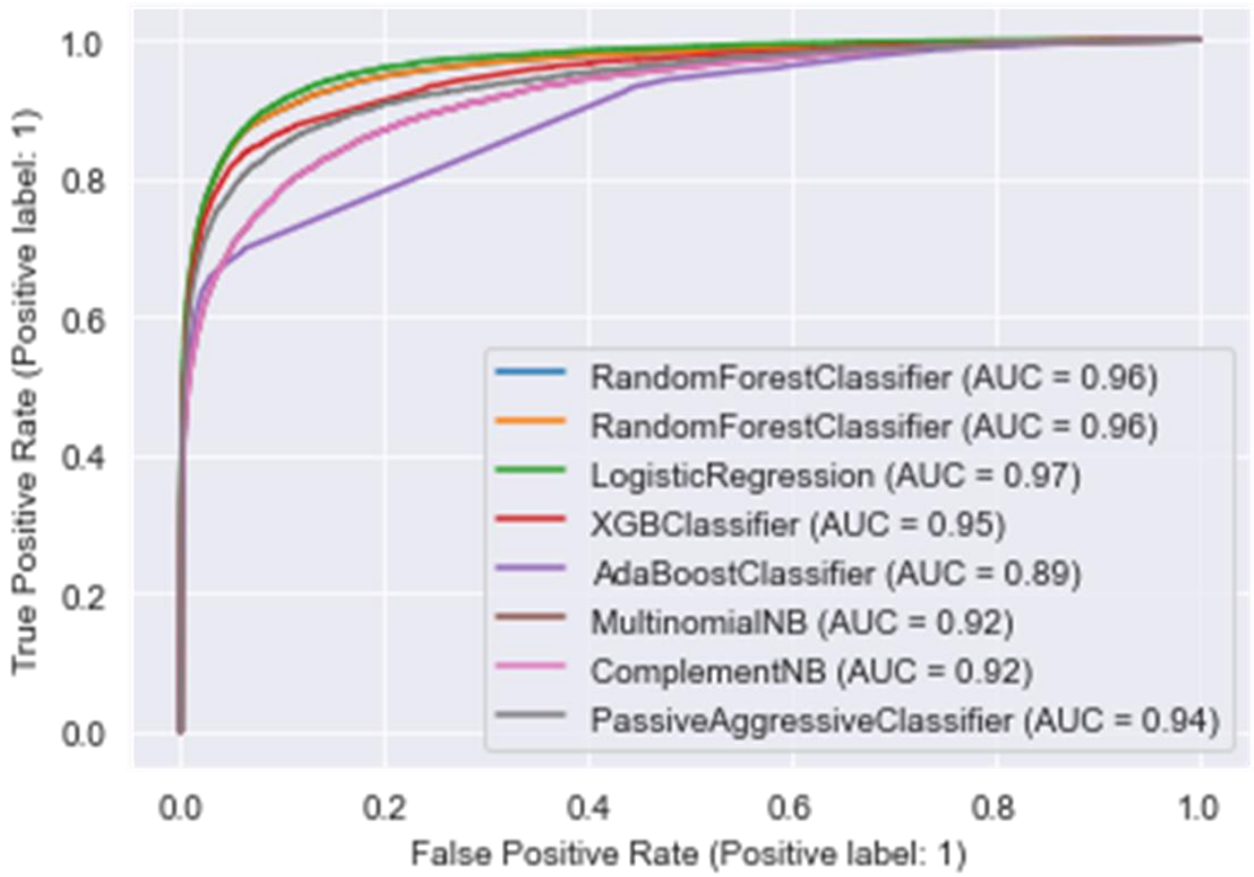
ROC AUC Scores

- The score is used to summarize the trade -off between the true positive rate and false positive rate for a predictive model using different probability thresholds. The AUC value lies between 0.5 to 1 where 0.5 denotes a bad classifier and 1 denotes an excellent classifier.



ROC AUC curves

the AUC-ROC curve helps us visualize how well our machine learning classifier is performing. ROC curves are appropriate when the observations are balanced between each class.



Logistic Regression ROC Curves

Interpretation of the Results

- Based on comparing the above graphs, roc_auc_scores, Precision, Recall, Accuracy Scores with Cross validation scores and log loss scores, it is determined that Random Forest Classifier and Logistic Regression are the best models for the dataset.

Hyper Parameter Tuning

GridSearchCV was used for Hyper Parameter Tuning of the Random Forest Classifier model.

RandomForestClassifier

```
params = {'n_estimators':[50,100,300],'max_depth': [10,60],'min_samples_leaf':[2,5,30],'min_samples_split':[1,2,5],'criterion':[
```

```
GridCV = GridSearchCV(RandomForestClassifier(),params,cv=5,n_jobs = -1,verbose = 1)
```

```
GridCV.fit(x_train,y_train)
```

```
Fitting 5 folds for each of 324 candidates, totalling 1620 fits

GridSearchCV(cv=5, estimator=RandomForestClassifier(), n_jobs=-1,
             param_grid={'criterion': ['gini', 'entropy'],
                         'max_depth': [10, 60],
                         'max_features': ['auto', 'sqrt', 'log2'],
                         'min_samples_leaf': [2, 5, 30],
                         'min_samples_split': [1, 2, 5],
                         'n_estimators': [50, 100, 300]},
             verbose=1)
```

```
GridCV.best_params_
```

```
{'criterion': 'gini',
 'max_depth': 60,
 'max_features': 'sqrt',
 'min_samples_leaf': 2,
 'min_samples_split': 5,
 'n_estimators': 100}
```

```
Best_mod = RandomForestClassifier(n_estimators = 300,criterion = 'gini', max_depth= 60, max_features = 'auto',min_samples_leaf =
Best_mod.fit(x_train,y_train)
rfpred = Best_mod.predict(x_test)
acc = accuracy_score(y_test,rfpred)
print(acc*100)
```

```
90.97593582887701
```

GridsearchCV was used for Hyper Parameter Tuning of the Logistic Regression model.

Logistic Regression

```
parameter = {'penalty':['l1', 'l2'],'dual': [True,False],'C':[1,2,5,10],'fit_intercept':[True,False],'solver':['liblinear','saga'
```

```
GridCV = GridSearchCV(LogisticRegression(),parameter,cv=5,n_jobs = -1,verbose = 1)
```

```
GridCV.fit(x_train,y_train)
```

```
Fitting 5 folds for each of 1024 candidates, totalling 5120 fits

GridSearchCV(cv=5, estimator=PassiveAggressiveClassifier(), n_jobs=-1,
             param_grid={'C': [1, 2, 5, 10], 'early_stopping': [True, False],
                         'fit_intercept': [True, False],
                         'max_iter': [1000, 1500, 2000, 5000],
                         'n_iter_no_change': [5, 10],
                         'tol': [0.001, 0.01, 0.1, 1.0],
                         'validation_fraction': [0.001, 0.1]},
             verbose=1)
```

```
GridCV.best_params_
```

```
{'C': 1,
 'early_stopping': False,
 'fit_intercept': True,
 'max_iter': 2000,
 'n_iter_no_change': 5,
 'tol': 0.1,
 'validation_fraction': 0.001}
```

```
Best_mod2 = LogisticRegression(penalty = 'l1',dual = False,fit_intercept=False,C=10,solver='liblinear',max_iter=100)
Best_mod2.fit(x_train,y_train)
LRpred = Best_mod2.predict(x_test)
acc = accuracy_score(y_test,LRpred)
print(acc*100)
```

```
93.88995655080214
```

Conclusion:

After Tuning the hyper parameters and based on the input parameter values and after fitting the train datasets it is found that Logistic Regression model performs the best.

The model was saved, and the Test Dataset was then prepared for final classification work by the model. This model was then tested using the Test Dataset. The model performed with good amount of accuracy.

- Key Findings and Conclusions of the Study

  The Model has 94.67% accuracy. But since the dataset was highly imbalanced that is not the best metric for measuring its efficiency. On the other hand, means that the model is optimized better to detect actual malignant comments. However, there is a need to strike a balance between precision and recall and have low false positives, which unnecessarily consume time and low false negatives which means only very few toxic comments deceive the model. F1 score of 0.96 provides a nuanced way to catch positive results without harming the usefulness of the model.

- Learning Outcomes of the Study in respect of Data Science

  The various data pre-processing and feature engineering steps in the project lent cognizance to various efficient methods for processing textual data. The NLTK suite is very useful in pre-processing text-based data and building classification models.

- Limitations of this work and Scope for Future Work

  The models were trained on a highly imbalanced dataset where the total malignant comments formed only 10% of the entire available data, which seriously affected the training and accuracy of the models. By training the models on more diverse data sets, longer comments, and a more balanced dataset, more accurate and efficient classification models can be built.

THANK YOU…