Project Report On

Malignant Comments Classifier Project

Prepared by:LAKSHMI PRANEETHA| Internship 26

ACKNOWLEDGEMENT

We would like to express our deep and sincere gratitude to FlipRobo for giving us the opportunity to do this project. As a great bridge between academic and industry, this program educated us how to perform theoretical methodology in real life. We would like to express our sincere thankfulness to my mentor Khushboo Garg, As our academic mentor, Khushboo Garg supported and helped in this project

INTRODUCTION

Business Problem Framing

The proliferation of social media enables people to express their opinions widely online. However, at the same time, this has resulted in the emergence of conflict and hate, making online environments uninviting for users. Although researchers have found that hate is a problem across multiple platforms, 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 behavior.

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.

Conceptual Background of the Domain Problem

In the past few years, it is seen that the cases related to social media hatred have increased exponentially. Social media is turning into a dark venomous pit for people nowadays. Online hate is the result of differences in opinion, race, religion, occupation, nationality etc. In social media the people spreading or involved in such kinds of activities use filthy language, 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 cyberbullying 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.

Online hate, described as abusive language, aggression, cyberbullying, hatefulness, insults, personal attacks, provocation, racism, sexism, threats, or toxicity has been identified as a major threat on online social media platforms. These kinds of activities must be checked for a better future.

Motivation for the Problem Undertaken

The main objective of this study is to investigate which method from a chosen set of machine learning techniques performs the best. So far, we have a range of publicly available models served through the Perspective API, including toxicity/malignant comments. But the current models still make errors, and they don't allow users to select which type of toxicity they are interested in finding.

The project gives an insight to identify major factors that lead to cyberbullying and online abusive comments. The exposure to real world data and the opportunity to deploy my skills in solving a real time problem has been the primary objective. However, the motivation for taking this project was that it is relatively a new field of research.

The main motivation was to classify the news in order to bring awareness and reduce unwanted chaos and make a good model which will help us to know such kinds of miscreants. Our goal is to build a prototype of online hate and abuse comment classifier which can be used to classify hate and offensive comments so that it can be controlled and restricted from spreading hatred and cyberbullying.

ANALYTICAL PROBLEM FRAMING

Mathematical/Analytical Modeling of the Problem

We are provided with two different datasets. One for training and another one to test the efficiency of the model created using the training dataset. The training data provided here has both dependent and independent variables. As it is a multiclass problem it has 6 independent/target variables. Here the target variables are named "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.

Clearly it is a binary classification problem as the target columns giving binary outputs and all independent variables have text so it is clear that it is a supervised machine learning problem where we can use the techniques of NLP and classification-based algorithms of Machine Learning. Here we will use NLP techniques like word tokenization, lemmatization, stemming and tfidf vectorizer then those processed data will be used to create best model using various classification based supervised ML algorithms like Logistic Regression, Multinomial NB, Extra Trees Classifier, XGBoost Classifier, LinearSVC, Decision Tree Classifier and Adaboost Classifier.

Data Sources & their formats

Data set provided by Flip Robo was in the format of CSV (Comma Separated Values). The data set contains the training set, which has approximately 159571 samples and the test set which contains nearly 153164 samples.

FEATURE INFORMATION:

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

Data preprocessing done

- Importing necessary libraries and loading the train and test datasets.
- Checked some statistical information like shape, number of unique values present, info, null values, value counts, duplicated values etc.
- Checked for null values and did not find any null values. And removed Id.
- Done feature engineering and created new columns viz 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.
- 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 a 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 cleaned data use TF-IDF vectorizer. It'll help to transform the text data to feature vectors which can be used as input in our modeling. It is a common algorithm to transform text into numbers. It measures the originality of a word by comparing the frequency of appearance of a word in a document with the number of documents the words appear in.Mathematically, TF-IDF = TF(t*d)*IDF(t,d)
- Balanced the data using Randomoversampler method.
- Proceeded with Model Building.

Data Inputs- Logic- Output Relationships

The dataset consists of a label and features. The features are independent and the label is dependent as the values of our independent variables change as our label varies.

- I checked the distribution of skewness using dist plots and used count plots to check the counts available in each column as a part of univariate analysis.
- To analyze the relation between features and label I have used many plotting techniques where I found some of the columns having strong relation with label.
- Got to know the sense of loud words in every label using word cloud which gives the words frequented in the labels.
- I have checked the correlation between the label and features using a heat map.

Hardware and Software Requirements & Tools used

To build the machine learning projects it is important to have the following hardware and software requirements and tools.

Hardware required:

- Processor: core i5 or above
- RAM: 8 GB or above
- ROM/SSD: 250 GB or above

Software required:

- Anaconda - language used Python 3

Libraries Used:

- import pandas as pd
- import numpy as np
- import seaborn as sns
- import matplotlib.pyplot as plt
- import warnings
  warnings.filterwarnings('ignore')
- 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
- from wordcloud import WordCloud
- from sklearn.model_selection import train_test_split
- from imblearn.over_sampling import RandomOverSampler
- from collections import Counter
- from sklearn import metrics
- from sklearn.model_selection import cross_val_score
- from sklearn.model_selection import train_test_split
- from sklearn.metrics import classification_report, confusion_matrix
- from sklearn.metrics importroc_curve, accuracy_score, roc_auc_score, hamming_loss, log_loss.
- from sklearn.linear_model import LogisticRegression
- from sklearn.tree import DecisionTreeClassifier
- from sklearn.svm import LinearSVC
- from sklearn.naive_bayes import MultinomialNB

- from sklearn.ensemble import AdaBoostClassifier
- from xgboost import XGBClassifier
- from sklearn.ensemble import ExtraTreesClassifier
- from sklearn.model_selection import cross_val_score
- from sklearn import datasets
- from sklearn import model_selection
- from sklearn.metrics import plot_roc_curve
- from sklearn.model_selection import GridSearchCV
- import joblib

MODEL BUILDING & EVALUATION

Identification of possible problem-solving approaches (methods)

I have used both statistical and analytical approaches to solve the problem which mainly includes the pre-processing of the data and also used EDA techniques and heat map to check the correlation of independent and dependent features. 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.

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".

I have used many classification algorithms and got the prediction results. By doing various evaluations I have selected the Logistic Regression Model as the best suitable algorithm to create our final model.

Testing of Identified Approaches (Algorithms)

Since the target variable is categorical in nature, from this I can conclude that it is a classification type problem hence I have used the following classification algorithms. After the pre-processing and data cleaning I left with 10 columns including targets.

The algorithms used on training the data are as follows:

1. Logistic Regression Model
2. Decision Tree Classifier Model
3. Linear SVC Model
4. MultinomialNB Classifier Model
5. AdaBoost Classifier Model
6. Extreme Gradient Boosting Classifier (XGB) Model
7. Extra Trees Classifier Model

Run and Evaluate selected models

Classification Models:

LOGISTIC REGRESSION:

In [75]:
```
from sklearn.linear_model import LogisticRegression

lg=LogisticRegression()
lg.fit(train_x,train_y)
pred_lg=lg.predict(x_test)

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

```
Accuracy score:  0.9446858288770054
Roc_auc_score:  0.895160256059955
Log loss :  1.9105144375628327
Hamming loss:  0.05531417112299465
Confusion matrix:
 [[41169  1835]
 [  813  4055]]
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       0.83      0.90      0.86     47872
weighted avg       0.95      0.94      0.95     47872
```

The Logistic Regression model gave us an Accuracy Score of 94.46 %.

DECISION TREE CLASSIFIER:

In [76]:
```
from sklearn.tree import DecisionTreeClassifier

dtc=DecisionTreeClassifier()
dtc.fit(train_x,train_y)
pred_dtc=dtc.predict(x_test)

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

```
Accuracy score:  0.929269719251337
Roc_auc_score:  0.8393066682299031
Log loss :  2.442971658720948
Hamming loss:  0.0707302807486631
Confusion matrix:
 [[40950  2054]
 [ 1332  3536]]
Classification Report:
                precision    recall  f1-score   support

           0       0.97      0.95      0.96     43004
           1       0.63      0.73      0.68      4868

    accuracy                           0.93     47872
   macro avg       0.80      0.84      0.82     47872
weighted avg       0.93      0.93      0.93     47872
```

The Decision Tree Classifier Model gave us an Accuracy Score of 92.92 %.

LINEAR SVC MODEL:

In [77]:
```
from sklearn.svm import LinearSVC

lsvc=LinearSVC()
lsvc.fit(train_x,train_y)
pred_lsvc=lsvc.predict(x_test)

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

```
Accuracy score:  0.9392337901069518
Roc_auc_score:  0.8845656195097403
Log loss :  2.098824158640169
Hamming loss:  0.06076620989304813
Confusion matrix:
 [[40991  2013]
 [  896  3972]]
Classification Report:
                precision    recall  f1-score   support

           0       0.98      0.95      0.97     43004
           1       0.66      0.82      0.73      4868

    accuracy                           0.94     47872
   macro avg       0.82      0.88      0.85     47872
weighted avg       0.95      0.94      0.94     47872
```

The Linear SVC Model gave us an accuracy score of 93.92 %.

MULTINOMIALNB CLASSIFIER MODEL:

In [78]:
```
from sklearn.naive_bayes import MultinomialNB

mnb=MultinomialNB()
mnb.fit(train_x,train_y)
pred_mnb=mnb.predict(x_test)

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

```
Accuracy score:  0.9107620320855615
Roc_auc_score:  0.8852045981161523
Log loss :  3.0822296316660607
Hamming loss:  0.0892379679144385
Confusion matrix:
 [[39447  3557]
 [  715  4153]]
Classification Report:
                precision    recall  f1-score   support

           0       0.98      0.92      0.95     43004
           1       0.54      0.85      0.66      4868

    accuracy                           0.91     47872
   macro avg       0.76      0.89      0.80     47872
weighted avg       0.94      0.91      0.92     47872
```

The MultinomialNB Classifier Model gave us an accuracy score of 91.07 %.

ADABOOST CLASSIFIER MODEL:

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
```

The AdaBoost Classifier Model gave us an accuracy score of 92.68 %.

XGBOOST CLASSIFIER MODEL:

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
```

The XGBoost Classifier Model gave us an accuracy score of 94.89 %.

EXTRA TREES CLASSIFIER MODEL:

```
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
```

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

From the above Classification Models, the highest accuracy score belongs to the Extra Trees Classifier, followed by the XGBoost Classifier and Logistic Regression Model.

Next, the Linear SVC Model followed by the AdaBoost Classifier and the Decision Tree Classifier.

Lastly the MultinomialNB Classifier Model.

Cross Validation Scores:

```
In [83]: scr_lg=cross_val_score(lg,x,y,cv=5,scoring='accuracy')
         print("Cross validation score of this model is: ",scr_lg.mean())
```

```
Cross validation score of this model is:  0.9559631724500548
```

The cross validation score of the Logistic Regression Model is 95.59 %.

```
In [84]: scr_dtc=cross_val_score(dtc,x,y,cv=5,scoring='accuracy')
         print("Cross validation score of this model is: ",scr_dtc.mean())
```

```
Cross validation score of this model is:  0.94047790017216
```

The cross validation score of the Decision Tree Classifier Model is 94.04 %.

```
In [85]: scr_lsvc=cross_val_score(lsvc,x,y,cv=5,scoring='accuracy')
         print("Cross validation score of this model is: ",scr_lsvc.mean())

         Cross validation score of this model is:  0.9592407120377594
```

The cross validation score of the Linear SVC Model is 95.92 %.

```
In [86]: scr_mnb=cross_val_score(mnb,x,y,cv=5,scoring='accuracy')
         print("Cross validation score of this model is: ",scr_mnb.mean())

         Cross validation score of this model is:  0.9463499000343936
```

The cross validation score of the MultinomialNB Classifier Model is 94.63 %.

```
In [87]: scr_abc=cross_val_score(abc,x,y,cv=5,scoring='accuracy')
         print("Cross validation score of this model is: ",scr_abc.mean())

         Cross validation score of this model is:  0.9457733566448472
```

The cross validation score of the AdaBoost classifier Model is 94.57 %.

```
In [88]: scr_xgb=cross_val_score(xgb,x,y,cv=5,scoring='accuracy')
         print("Cross validation score of this model is: ",scr_xgb.mean())

         Cross validation score of this model is:  0.9536005912254529
```

The cross validation score of the XGBoost Classifier Model is 95.36 %.

From the above Cross Validation Scores, the highest CV score belongs to the LinearSVC model, Logistic Regression Model.
Next the XGBoost Classifier model , the MultinomialNB Classifier and the AdaBoost Classifier Model.
Lastly, the Decision Tree Classifier.

Hyper Parameter Tuning:

Since the Accuracy score and the cross validation score of the Logistic Regression Model are good and the AUC score is the highest among others we shall consider this model for hyper parameter tuning.
We shall use GridSearchCV for 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'}
```

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
```

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

Saving the final model and predicting the saved model

Now we shall save the best model.

```
# Saving the model using .pkl
import joblib
joblib.dump(Final_Model,"Malignant_Comments_Classifier.pkl")
```

96]: ['Malignant_Comments_Classifier.pkl']

```
# Loading the final model
model = joblib.load('Malignant_Comments_Classifier.pkl')
```

```
# Lets load the test data set
test_mc
```

Predictions from the saved model.

```
# Lets load the test data set
test_mc
```

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

153164 rows × 4 columns

```
# Predicting the values for test data after loading trained model
Predictions = model.predict(x1)
Predictions
```

```
array([0, 0, 0, ..., 0, 0, 0])
```

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

	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

```
# Checking values counts for predicted values
test_mc.Predicted_Values.value_counts()
```

```
0    146573
1      6591
Name: Predicted_Values, dtype: int64
```

```
# Saving the data into csv file
test_mc.to_csv("Malignant_Comments_Classifier_Predicted_Test_Data.csv",index=False)
```

Key Metrics for success in solving problem under consideration

I have used the following metrics for evaluation:

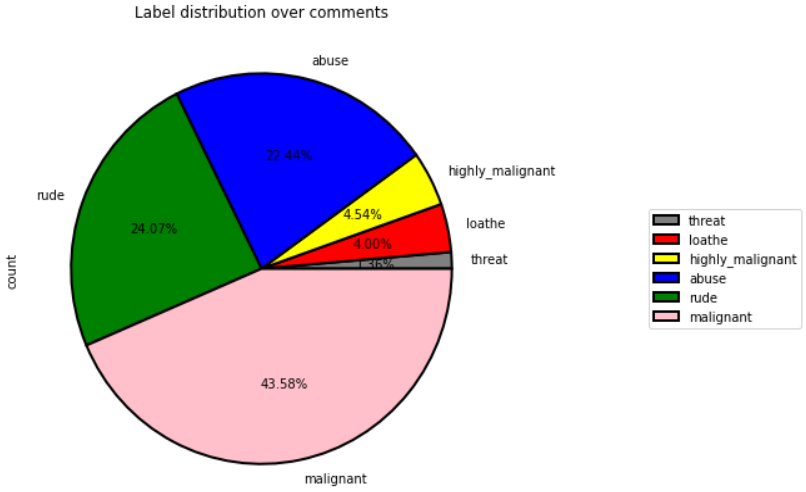
- Accuracy score, which is used when the True Positives and True negatives are more important.
- Confusion Matrix, which is a matrix used to determine the performance of the classification models for a given set of test data.
- Classification report, which is used to measure the quality of predictions from a classification algorithm.
- Cross Validation Score, which is a technique for evaluating ML models by training several ML models on subsets of the available input data and evaluating them on the complementary subset of the data.
- Precision is the degree to which repeated measurements under the same conditions are unchanged. It is the amount of information that is conveyed by a value. It refers to the data that is correctly classified by the classification algorithm.
- Recall is how many of the true positives were recalled (found). Recall refers to the percentage of data that is relevant to the class.
- F1 Score is used to express the performance of the machine learning model (or classifier). It gives the combined information about the precision and recall of a model. This means a high F1-score indicates a high value for both recall and precision.

- Log Loss is the most important classification metric based on probabilities. Log Loss is the negative average of the log of corrected predicted probabilities for each instance.
- Hamming Loss is the fraction of wrong labels to the total number of labels. In multi-class classification, hamming loss is calculated as the hamming distance between y_true and y_pred. In multi-label classification, hamming loss penalizes only the individual labels.

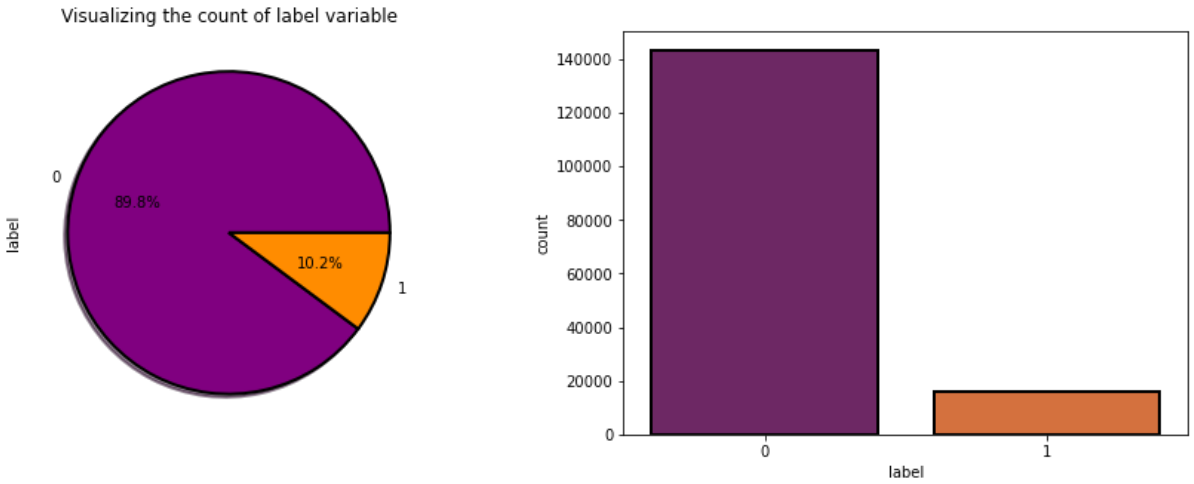
Visualizations

I used pandas profiling to get the over viewed visualization on the pre-processed data. I have used various plots for visualization such as pie charts, count plots, distribution plots to help understand the features. I have used word clouds to get the sense of loud words in the labels.

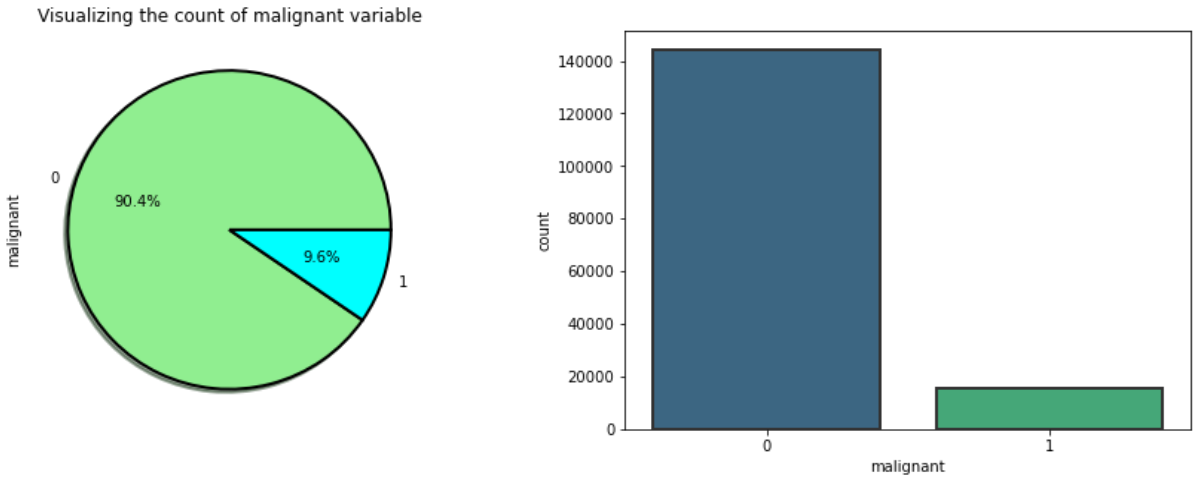
PLOTS:



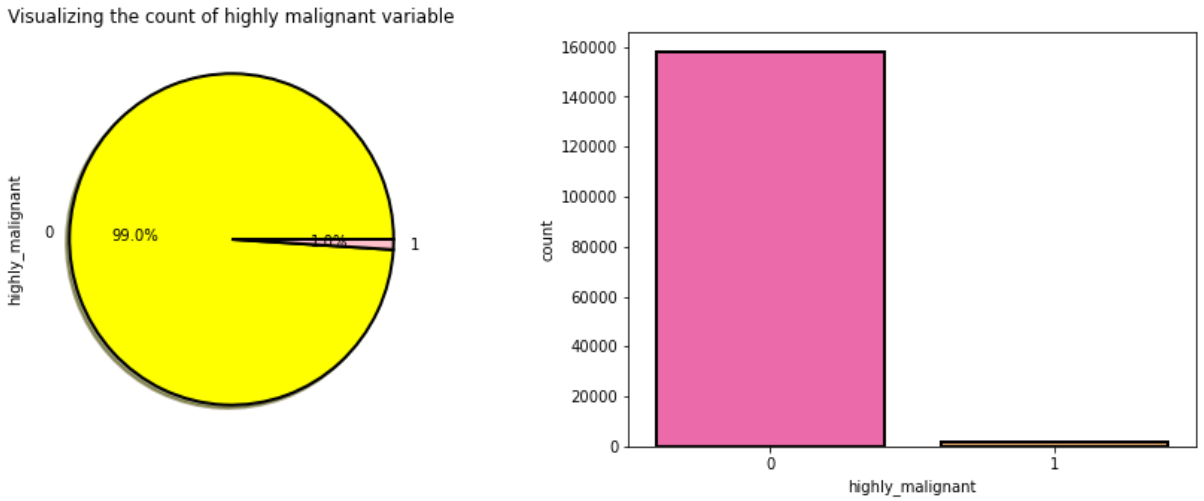
From the pie chart we can notice approximately 43.58 % of the comments are malignant, 24.07 % of the comments are rude and 22.44 % are of abuse. The count of malignant comments are high compared to other types of comments and the count of threat comments are very less.



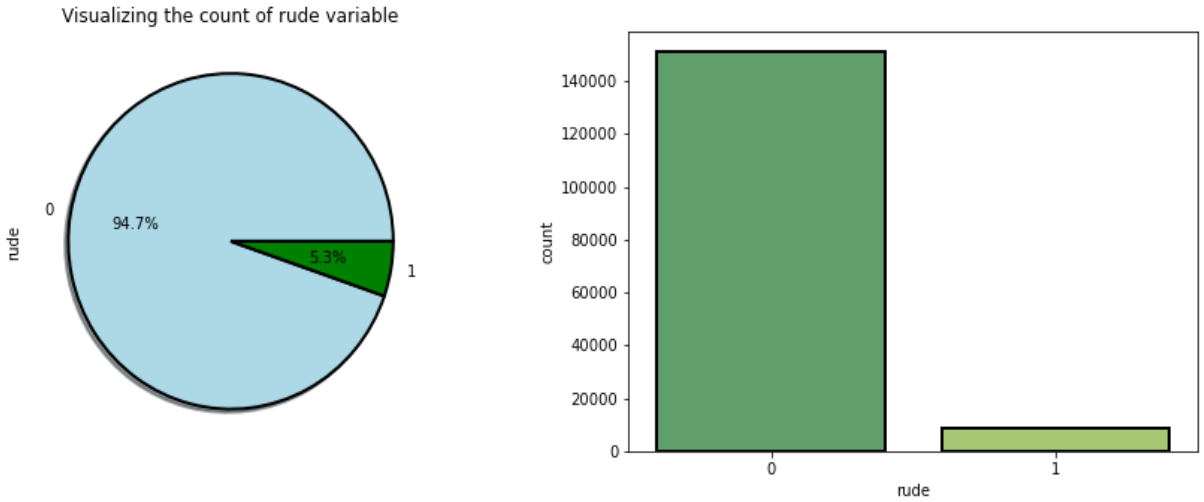
From the above plots we can observe the count of negative comments are high compared to the non negative comments. Here around 89.8% of the comments are turned out to be negative comments and only 10.2 % of them are considered to be positive or neutral comments. We can also observe the data imbalance issue here, we need to balance the data.



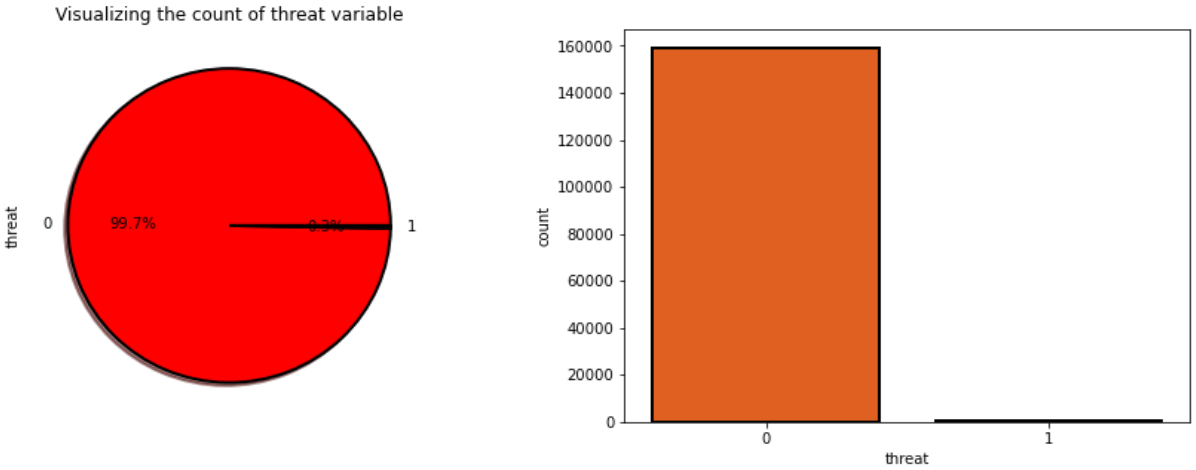
From the above plots we can observe the count of malignant comments is high compared to non malignant comments. That is around 90.4 % of the comments are malignant and only 9.6 % of the comments are good.



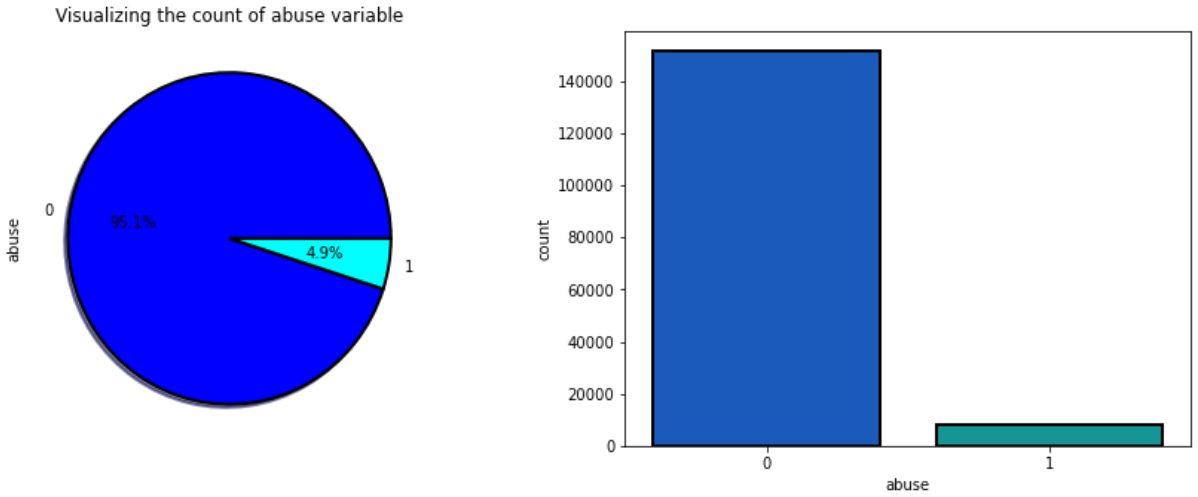
From the plot we can observe that the count of highly malignant comments is very high, which is about 99 % and only 1 % of the comments are normal.



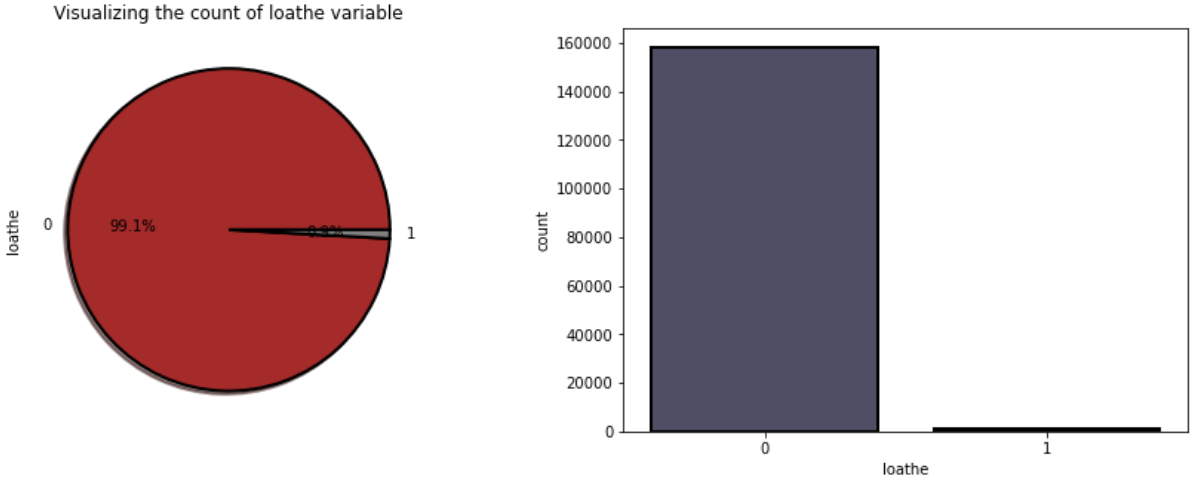
The number of rude comments are high compared to normal comments. Around 94.7 % of the comments fall into rude and remaining comments are considered to be normal comments.



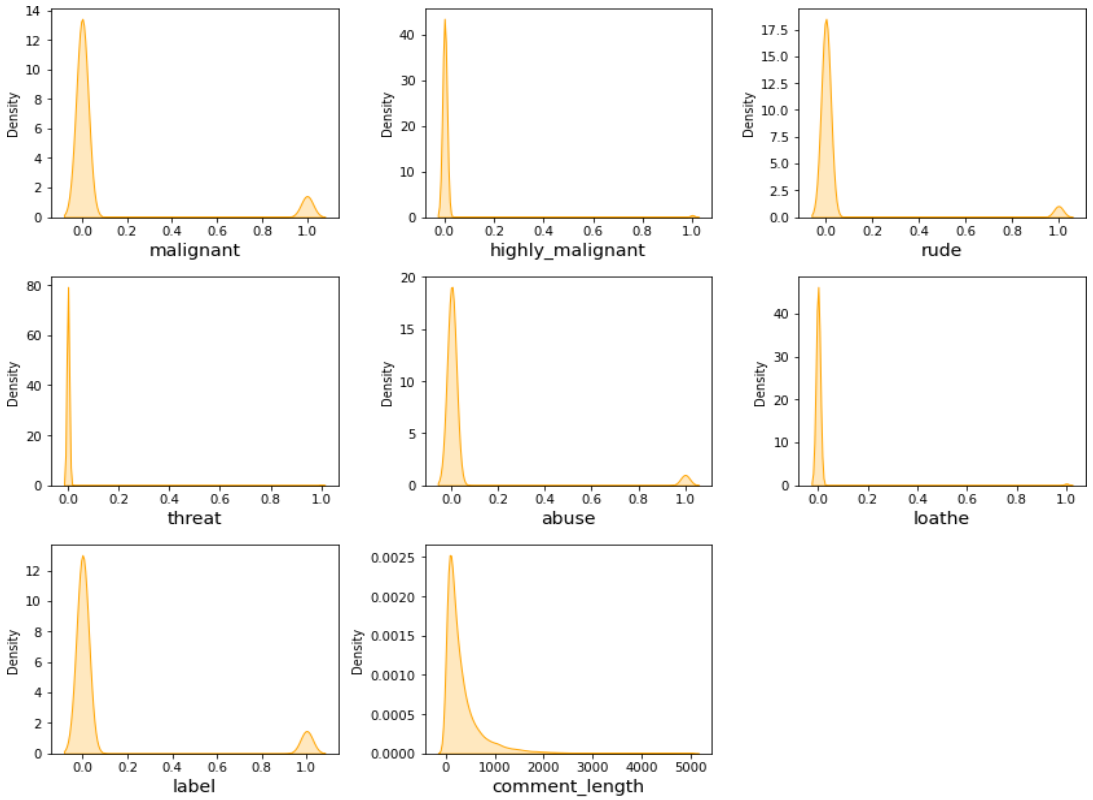
In the above visualization also, 99.7 % of the comments are threatening and only 0.3 % of the comments look normal.



The count of abusing type comments is high which is 95.1 % and only 4.9 % of the comments are normal.

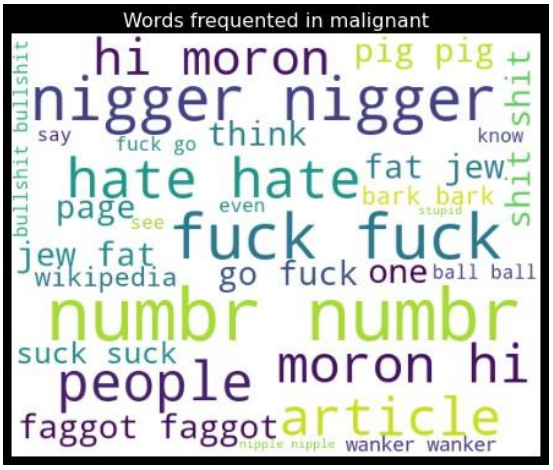


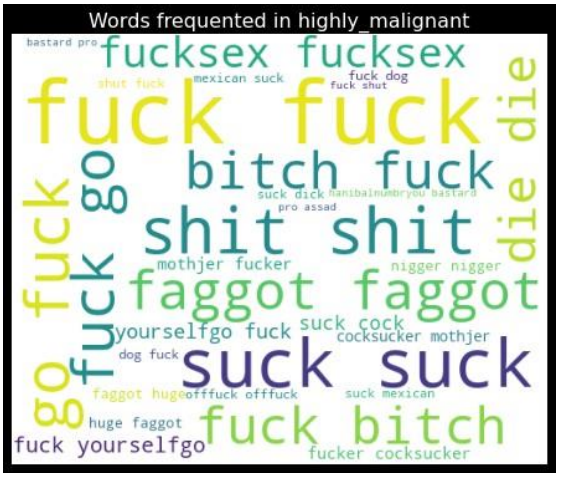
The count of loathe is high (99.1 %) compared to normal (0.9 %) text comments.

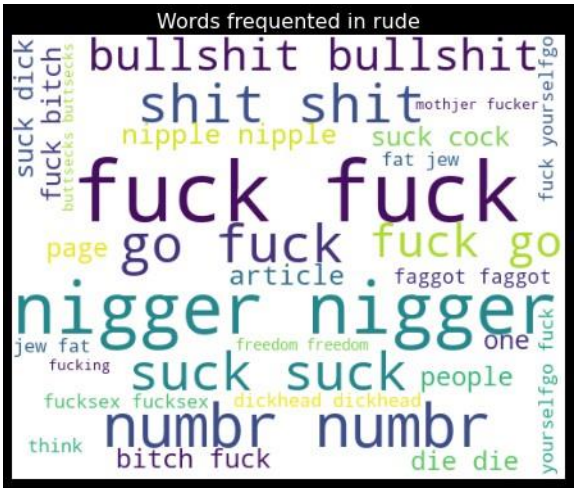


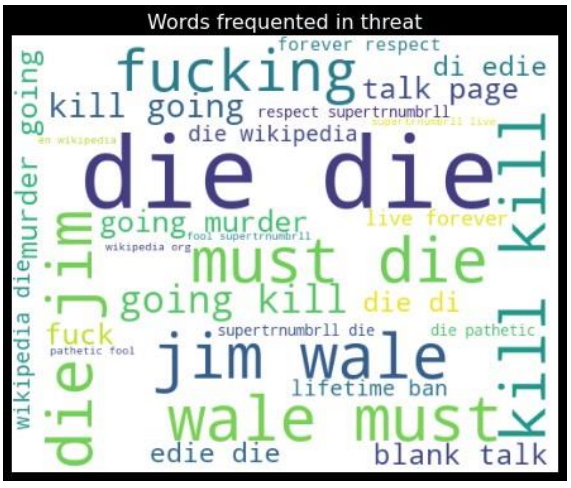
From the distribution plots we can notice that all the columns are skewed to the right except the comment label column. Since all the columns are categorical in nature there is no need to remove skewness and outliers in any of the columns.

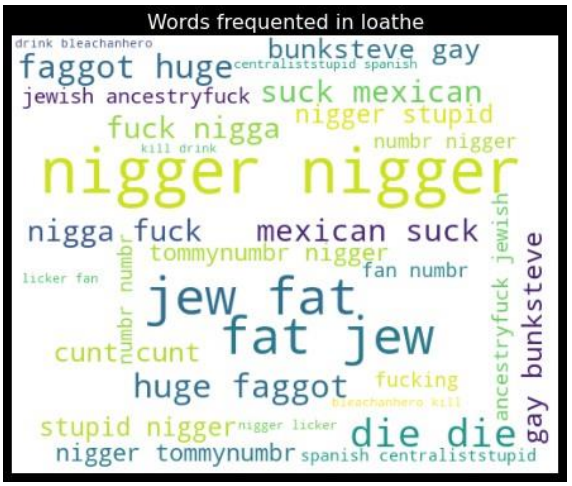
WORD CLOUDS:











From the above plots we can clearly see the toxic words which are an indication of malignant, highly malignant, rude, threat, abuse and loathe words. Here most frequent words used for each label are displayed in the word cloud based on different labels and also when all the values are present.

Interpretation of the Results

Visualizations: I have used a distribution plot to visualize how the data has been distributed. Used count plots and pie charts to check the count of a particular category for each feature. The heat map helped me to understand the correlation between dependent and independent features. Also, heat maps helped to detect the multicollinearity problem and feature importance. With the help of Word Clouds I would be able to sense the loud words in each label. The AUC-ROC curve helped to select the best model.

Pre-processing: The dataset should be cleaned and scaled to build the ML models to get good predictions. I have performed a few NLP text processing steps which I have already mentioned in the pre-processing steps where all the important features are present in the dataset and ready for model building.

Model building: After cleaning and processing data, I performed a train test split to build the model. I have built multiple classification models to get the accurate accuracy score, and evaluation metrics like precision, recall, confusion matrix, f1 score, log loss, hamming loss. I got the Logistic Regression Model as the best model which gives a 94.46% accuracy score. I checked the cross-validation score ensuring there will be no overfitting. After tuning the best model, I got a 94.49% accuracy score. Finally, I saved my final model and got the good predictions results for the test dataset.

CONCLUSION

Key Findings and Conclusions of the Study

- From the above analysis the below mentioned results were achieved which depicts the chances and conditions of a comment being a hateful comment or a normal comment;
- With the increasing popularity of social media more and more people consume feeds from social media and due differences they spread hate comments instead of love and harmony. It has strong negative impacts on individual users and broader society.
- From this dataset we were able to understand the impact of various malignant comments, the different types of malignant comments and how to identify them.
- In this study, we have used multiple machine learning models to predict malignant comments and identify them.
- We have gone through the data analysis by performing feature engineering, finding the relation between features and the label through visualizations. And got the important features and we used these features to predict the defaulters' rate by building ML models.
- After training the model we checked CV score to overcome the overfitting issue.
- Performed hyper parameter tuning, on the best model. We have also got good prediction results.

Learning Outcomes of the Study in respect of Data Science

While working on this project I learned many things and gained new techniques and ways to deal with uncleaned text data. Found how to deal with multiple target features. Tools used for visualizations give a better understanding of the dataset. We have used a lot of algorithms and find that in the classification problem where we have only two labels, the Logistic Regression Model gives better results compared to others.

THANKYOU