MALIGNANT COMMENTS CLASSIFICATION

USING NLP

Submitted by:

VANISREE P G

ACKNOWLEDGMENT

First I would like to thank the Almighty for his wonderful presence with me throughout this project and helped me to make it as a successful one.

For my internship I had the pleasure of working at FILP ROBO Was a great chance for acquired knowledge, personal and Professional development.

I extend whole hearted thanks to FILP ROBO under whom I worked and learned a lot and for enlightening me with their knowledge and experience to grow with the corporate working.

This is a great pleasure to express my deep sense of gratitude and thanks to SME for his valuable ideas, instantaneous help, effective support and continued encouragement which enabled for the successful completion of the project. I also like to thank the data trained mentors and Technical team members for helping me with technical queries.

And these are the following website which I referred for the reference

1. https://www.kaggle.com/
2. https://scikit-learn.org/
3. www.stackoverflow.com
4. www.google.com
5. www.geeksforgeeks.org

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, cyber bullying, 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.

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 un offensive, but “u are an idiot” is clearly offensive.

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 cyber bullying.

➢ Conceptual Background of the Domain Problem

Online platforms when used by normal people can only be comfortably used by them only when they feel that they can express themselves freely and without any reluctance. If they come across any kind of a malignant or toxic type of a reply which can also be a threat or an insult or any kind of harassment which makes them uncomfortable, they might defer to use the social media platform in future. Thus, it becomes extremely essential for any organization or community to have an automated system which can efficiently identify and keep a track of all such comments and thus take any respective action for it, such as reporting or blocking the same to prevent any such kind of issues in the future.

This is a huge concern as in this world, there are 7.7 billion people, and, out of these 7.7 billion, more than 3.5 billion people use some or the other form of online social media. Which means that every one-in-three people uses social

media platform. This problem thus can be eliminated as it falls under the category of Natural Language Processing. In this, we try to recognize the intention of the speaker by building a model that's capable of detecting different types of toxicity like threats, obscenity, insults, and identity-based hate. Moreover, it is crucial to handle any such kind of nuisance, to make a more user-friendly experience, only after which people can actually enjoy in participating in discussions with regard to online conversation.

➢ Review of Literature

The main goal of the challenge is developing a multi-label classifier, not only to identify the toxic comments but also detect the type of toxicity such as threats, obscenity, insults, and identity-based hate; while, there is still no practical tools that is able to distinguish these types of comments from each other with a very low rate of errors and a high accuracy. Automatic recognition of toxic contents in online forums, and social media is 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 comment posted online motivated the current study. As mentioned above, distinguishing improper comment that brings harassment to privacy and morality is one of the most crucial subjects related to the current extension of social media. With all the progress and improvement in IT and data science, the world is in requirement of a properly designed technique to find and

isolate these kinds of comments that we call it toxic. Accordingly, aiming to find and develop an efficient algorithm to identify toxic comments, the current study implemented several algorithms, including Naıve Bayes (as a benchmark), Recurrent Neural Network (RNN), and Long Short Term Memory (LSTM). In order to achieve our goal in this study, related comments data of Crowd-sourcing (159,000 text comments) that was formerly labeled as seven categories were used. Our contributions in this paper demonstrate that a practical data science solution is more than an algorithm, and that the scalability of the architecture must be considerate of the task at hand. First, we simplified the challenge to a binary classification whether the comment is toxic or not toxic. We also demonstrate that utilizing elastic cloud computing resources provides the scalability necessary to provide a practical solution. Finally, within this framework, we employed an advanced deep learning algorithm that provided a significant improvement in classification, versus our baseline modelling.Past helpfulness records tends to predict future helpfulness ratings. Finally they conclude that characteristics of reviewers and review messages have a varying degree of impact on review helpfulness. Most of the previous works either use vector space model or traditional semantic analysis with document features to improve the prediction. Considering that in mind, we use several linguistic(lexical, semantic), anatomical, and metadata information with word embedding for feature extraction

➢ Motivation for the Problem Undertaken

To understand real world problems where Machine Learning and Data Analysis can be applied to help to predict the prices in various domains to make better decisions with the help of which they can gain profit or can be escaped from any loss which otherwise could be possible without the study of data. The project has the provided by the Flip Robo Technologies as a part of the internship programme .This project help to exposure to real world data .To display my skill in solving a real time problem has been the motivation.

For preparing the desired data a simple code was written in python to remove the useless features. Many features were removed except the summary of the review, the text of the review itself, score and product. The score that is generated by the reviewer includes a number of stars on scales of 1 to 5. Reviews that were rated with one or two stars were considered as negative and those with four or five stars were considered as positive. Reviews with three stars usually contain many mixed reviews and are difficult to be labelled into a positive or negative category.

Analytical Problem Framing

➢ Mathematical/ Analytical Modelling of the Problem

In the whole research process various mathematical, statistical and analytics modelling has been done. The background for the problem originates from the multitude of online forums, where-in people participate actively and make comments. As the comments some times may be abusive, insulting or even hate-based, it becomes the responsibility of the hosting organizations to ensure that these conversations are not of negative type. The task was thus to build a model which could make prediction to classify the comments into various categories.

Our proposed approach is depicted that consists of five major steps. Firstly, Given a group of sentences or paragraphs, used as a comment by a user in an online platform, classify it to belong to one or more categoriesfed into the feature generation engine that includes four different functions such as lexical, structural, semantic and combined. Secondly, generated features are used as input to standard machine learning classification algorithm. In the third step, four different trained

Models are built using the classification algorithm. In the fourth and fifth steps, test reviews are evaluated based on the pre-trained classification models for the comment is malignant or not. We consider the - Malignant helpfulness prediction as a binary classification

problem once -Malignant are labeled as helpful.As the task was to figure out whether the data belongs to zero, one, or more than one categories out of the six listed above, the first step before working on the problem was to distinguish between multi-label and multi-class classification.

In multi-class classification, we have one basic assumption that our data can belong to only one label out of all the labels we have. For example, a given picture of a fruit may be an apple, orange or guava only and not a combination of these.

In multi-label classification, data can belong to more than one label simultaneously. For example, in our case a comment may be toxic, obscene and insulting at the same time. It may also happen that the comment is non-toxic and hence does not belong to any of the six labels.

Hence, I had a multi-label classification problem to solve. The case of multi-label is therefore not applicable to us. In order to apply text classification, the unstructured format of text has to be converted into a structured format for the simple reason that it is much easier for computer to deal with numbers than text. This is mainly achieved by projecting the textual contents into Vector Space Model, where text data is converted into vectors of numbers. In the field of text classification, documents are commonly treated like a Bag-of-Words (BoW),

meaning that each word is independent from the others that are present in the document. They are examined without regard to grammar neither to the word order4 . In such a model, the term-frequency (occurrence of each word) is used as a feature in order to train the classifier. However, using the term frequency implies that all terms are considered equally important. As its name suggests, the term frequency simply weights each term based on their occurrence frequency and does not take the discriminatory power of terms into account. To address this problem and penalize words that are too frequent, each word is given a term frequency inverse document frequency (tf-idf) score Flesch reading ease

➢ Data Sources and their formats

The data set contains the training set, which has approximately 1,59,000 sa mples and the test set which contains nearly 1,53,000 samples. All the data samples contain 8 fields which includes 'Id', 'Comments', 'Malignant', 'Hi ghly malignant', 'Rude', 'Threat', 'Abuse' and 'Loathe'.

The label can be either 0 or 1, where 0 denotes a NO while 1 denotes a YE S. There are various comments which have multiple labels. The first attribu te is a unique ID associated with each comment.

The Train data set includes:

- **Malignant**: It is the Label column, which includes values 0 and 1, deno ting 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 threa t to someone.
- **Abuse**: It is for comments that are abusive in nature.
- **Loathe**: It describes the comments which are hateful and loathing in nat ure.

b)Test dataset

- **ID**: It includes unique Ids associated with each comment text given.
- **Comment text**: This column contains the comments extracted from vari ous social media platforms. I will start by importing all the necessary librar ies that we need for this task and import the dataset.

1) Importing libraries

2) Importing the dataset

Our target is to find the insights of the data and to do thorough data analysis.

```
In [1]: import pandas as pd
        import numpy as np
        import matplotlib as mlp
        import seaborn as sns
        import matplotlib.pyplot as plt
        import warnings
        warnings.filterwarnings('ignore')

In [2]: trn_data=pd.read_csv(r"C:\Users\Hi\Downloads\Malignant-Comments-Classifier-Project\Malignant Comments Classifier Project\train.c

In [3]: trn_data

Out[3]:
```

	id	comment_text	malignant	highly_malignant	rude	threat	abuse	loathe
0	0000997932d777bf	Explanation\nWhy the edits made under my usern...	0	0	0	0	0	0
1	000103f0d9cfb60f	D'aww! He matches this background colour I'm s...	0	0	0	0	0	0
2	000113f07ec002fd	Hey man, I'm really not trying to edit war. It...	0	0	0	0	0	0
3	0001b41b1c6bb37e	"\nMoreover\nI can't make any real suggestions on ...	0	0	0	0	0	0
4	0001d958c54c6e35	You, sir, are my hero. Any chance you remember...	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...
159566	ffe987279560d7ff	":::::And for the second time of asking, when ...	0	0	0	0	0	0

```
In [9]: #the complete information about the dataset
        trn_data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 159571 entries, 0 to 159570
Data columns (total 7 columns):
 #   Column             Non-Null Count   Dtype
---  ------             --------------   -----
 0   comment_text       159571 non-null  object
 1   malignant          159571 non-null  int64
 2   highly_malignant   159571 non-null  int64
 3   rude               159571 non-null  int64
 4   threat             159571 non-null  int64
 5   abuse              159571 non-null  int64
 6   loathe             159571 non-null  int64
dtypes: int64(6), object(1)
memory usage: 8.5+ MB
```

```
In [10]: trn_data.describe()
```

Out[10]:

	malignant	highly_malignant	rude	threat	abuse	loathe
count	159571.000000	159571.000000	159571.000000	159571.000000	159571.000000	159571.000000
mean	0.095844	0.009996	0.052948	0.002996	0.049364	0.008805
std	0.294379	0.099477	0.223931	0.054650	0.216627	0.093420
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
50%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
75%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

➢ Data Pre-processing

Before building model, the data should be properly pre processed and converted to quality, clean data even the resulting machine learning model will be of great quality .The data pre-processing includes three main parts that is data integration, data cleaning, data transformation. In data integration the data collected from various sources are integrated. In data cleaning process the data containing the null values, unnecessary rows with null values are being cleared. The data transformation includes the feature scaling ,categorical data, etc to set the certain range of data.

➢ The data set contains the training set, which has approximately 1,59,000 samples and the test set which contains nearly 1,53,000 samplesRating is our dependent variable.

a) Checking for duplicates and dropping them

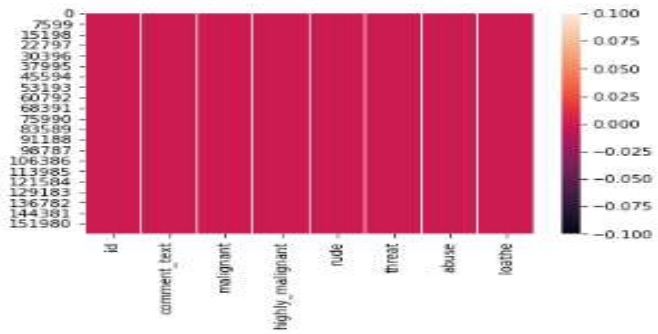
```
In [5]: trn_data.duplicated().sum()
Out[5]: 0
```

There is no duplicated dataset

b) **Checking missing value from the data set.**

```
In [6]: trn_data.isnull().sum()
Out[6]: id                0
        comment_text      0
        malignant         0
        highly_malignant  0
        rude              0
        threat            0
        abuse             0
        loathe            0
        dtype: int64
```

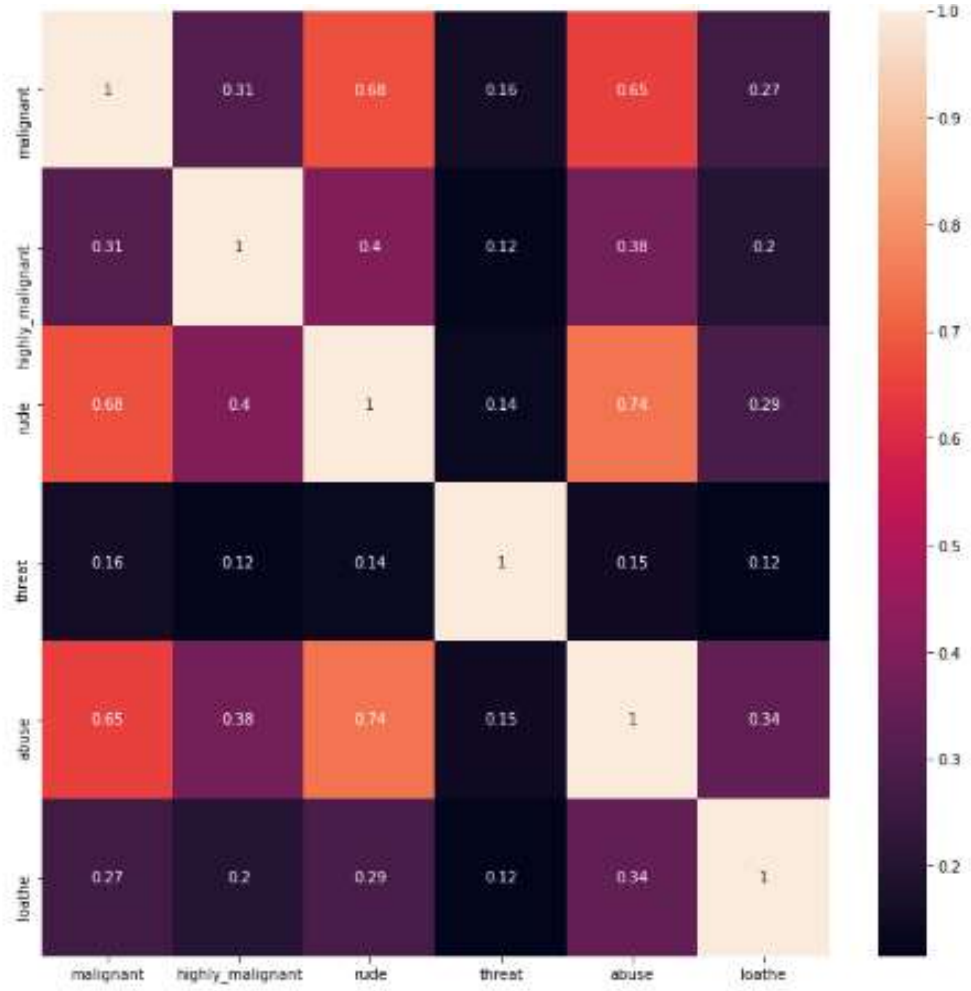
```
In [7]: import seaborn as sns
        sns.heatmap(trn_data.isnull())
Out[7]: <AxesSubplot:>
```



- Data contain no missing value

Correlation

- The highest positive correlation is seen in between fields 'rude' and 'abuse'.
- Attribute 'threat' is negatively correlated with each and every other feature of this training dataset.
- Almost all variable are correlated with each other negatively.



Data is pre-processed using following technique:

a) Removing punctuation

b) Removing numbers

c) Converting text to lower case (no capital letters)

d) Removing extra whitespace

e) Removing stop-words (extremely common words which do not provide any analytic information and tend to be of little value i.e. a, and, are etc.)

f) Tokenization

g) Stemming

h) Lemmatization

i) Apply Text vectorization to convert text to numeric

➢ Data Inputs- Logic- Output Relationships

The dataset consists of 8 features columns. 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. Using word cloud, we can see most occurring word for different categories.

To gain more insight about relationship between input & output heatmap of correlation and bar plot of correlation of label with independents features is plotted.

➢ Hardware and Software Requirements and Tools Used

Hardware used for doing the project is a ‘Laptop’ with high end specification and stable internet connection .while coming to the software part I had used ‘python jupyter notebook’ for do my python program and data analysis.

Excel file and Microsoft excel are required for the data handling. In jupyter notebook I had imported lot of python libraries are carried to this project.

1.Pandas-a library which is used to read the data ,visualisation and analysis of data.

2.Numpy-used for working with array and various mathematical operations in python.

3.Seaborn- visualization for plotting different type of plot.

4.Matplotlib- It provides an object-oriented API for embedding plots into applications .

5.selenium- web scraping the data.

6.Libraries used for text mining/text analysis are nltk, stopword, WorldLemmatizer ,tfidvectorizer,Word_tokenize.

6. Libraries used for Machine learning model bulding like LogisticRegression, SVM, RandomForestClassifer.

Model/s Development and Evaluation

➢ Identification of possible problem-solving approaches (methods)

In machine learning, several algorithms are applied to predict rating on review. The algorithms are: Linear regression, Decision tree, SVR, Gradient Boosting Regression, Ridge and Random Forest Algorithm. These models have been implemented using the python library Sklearn. The parameters like accuracy score , confusion matrix Cross Validation Score are considered to check the efficiency of these models. Further Hyper tuning to build more accurate model

Regression Model with following algorithms

- Logistic Regression
- Decision Tree Regression
- SVR

Evaluation metrics

- Accuracy Score
- Confusion matrix
- Cross val_ score.

Testing of Identified Approaches (Algorithms)

Listing down all the algorithms used for the training and testing.

- LR=LinearRegression()
- DT=DecisionTreeRegressor()
- svr=SVR()

3. KEY METRICS FOR SUCCESS IN SOLVING PROBLEM UNDER CONSIDERATION

Precision can be seen as a measure of quality; higher precision means that an algorithm returns more relevant results than irrelevant ones.

▪Recall is used as a measure of quantity and high recall means that an algorithm returns most of the relevant results.

▪Accuracy score is used when the True Positives and True negatives are more important. Accuracy can be used when the class distribution is similar.

▪F1-score is used when the False Negatives and False Positives are crucial. While F1-score is a better metric when there are imbalanced classes.

Cross validation Score: To run cross-validation on multiple metrics and also to return train scores, fit times and score times. Get predictions from each split of cross-validation for diagnostic purposes. Make a scorer from a performance metric or loss function.

▪AUC_ROC _score: ROC curve. It is a plot of the false positive rate (x-axis) versus the true positive rate (y-axis) for a number of different candidate threshold values between 0.0 and 1.0

▪We have used Accuracy Score and Cross validation score as key parameter for model evaluation in this project since balancing of data is perform.

Run and Evaluate selected models

Describe all the algorithms used along with the snapshot of their code and what were the results observed over different evaluation metrics. Train-test data splits were conducted. In this situation, we split the data into training and test sets, then fit candidate models on the training set, evaluate and select them on the test set.

1.Logistic Regression

In [30]:
```
# Creating train_test_split using best random_state
x_train, x_test, y_train, y_test = train_test_split(X, Y, random_state=88, test_size=.3)
```

In [36]:
```
from sklearn.linear_model import LogisticRegression
# creating the model
model_lg = LogisticRegression()

# feeding the training set into the model
model_lg.fit(x_train, y_train)

# predicting the results for the test set
y_pred = model_lg.predict(x_test)

# calculating the training and testing accuracies
print("Training accuracy :", model_lg.score(x_train, y_train))
print("Testing accuracy :", model_lg.score(x_test, y_test))
#Accuracy Score
print('Accuracy Score of Logistics Regression :', accuracy_score(y_test, y_pred))
# classification report
print('classification Report of Logistics Regression :\n',classification_report(y_test, y_pred))

# confusion matrix
print('Confusion matrix of Logistics Regression :\n',confusion_matrix(y_test, y_pred))
```

```
Training accuracy : 0.9581643524113913
Testing accuracy : 0.9559241310160428
Accuracy Score of Logistics Regression : 0.9559241310160428
classification Report of Logistics Regression :
              precision    recall  f1-score   support

           0       0.96      1.00      0.98     43111
           1       0.94      0.60      0.73      4761

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

Confusion matrix of Logistics Regression :
 [[42914   197]
 [ 1913  2848]]
```

In [37]:
```
from sklearn.model_selection import cross_val_score
CVscore = cross_val_score(model_lg, X, Y, cv =5)
print('\033[1m'+'Cross Validation Score', model_lg, ':'+'\033[0m\n')
print("CVScore :" ,CVscore)
print("Mean CV Score :",CVscore.mean())
print("Std deviation :",CVscore.std())
```

```
Cross Validation Score LogisticRegression() :

CVScore : [0.95447282 0.95425285 0.95321882 0.95447139 0.95343736]
Mean CV Score : 0.9539730320806239
Std deviation : 0.0005353399282923333
```

2.Decision Tree Classifier

In [56]:
```
from sklearn.tree import DecisionTreeClassifier

# creating model
model_dt = DecisionTreeClassifier()

# feeding the training set into the model
model_dt.fit(x_train, y_train)

# predicting the results for the test set
y_pred = model_dt.predict(x_test)

# calculating the training and testing accuracies
print("Training accuracy :", model_dt.score(x_train, y_train))
print("Testing accuracy :", model_dt.score(x_test, y_test))

#Accuracy Score
print('Accuracy Score of DecisionTreeClassifier :', accuracy_score(y_test, y_pred))

# classification report
print('classification Report of DecisionTreeClassifier :\n',classification_report(y_test, y_pred))

# confusion matrix
print('Confusion matrix of DecisionTreeClassifier :\n',confusion_matrix(y_test, y_pred))
```

```
Training accuracy : 0.9997403736828441
Testing accuracy : 0.9441844919786097
Accuracy Score of DecisionTreeClassifier : 0.9441844919786097
classification Report of DecisionTreeClassifier :
               precision    recall  f1-score   support

           0       0.97      0.97      0.97     43111
           1       0.73      0.70      0.71      4761

    accuracy                           0.94     47872
   macro avg       0.85      0.84      0.84     47872
weighted avg       0.94      0.94      0.94     47872

Confusion matrix of DecisionTreeClassifier :
 [[41860  1251]
 [ 1421  3340]]
```

In [57]:
```
from sklearn.model_selection import cross_val_score
CVscore = cross_val_score(model_dt, X, Y, cv =3)
print('Cross Validation Score', model_dt, ':\n')
print("CVScore :" ,CVscore)
print("Mean CV Score :",CVscore.mean())
print("Std deviation :",CVscore.std())
```

```
Cross Validation Score DecisionTreeClassifier() :

CVScore : [0.94279107 0.94149276 0.94363602]
Mean CV Score : 0.9426399524279395
Std deviation : 0.0008814832549460678
```

3.SVC

In [48]:
```
from sklearn.svm import SVC

# creating the model
model_svc = SVC()

# feeding the training set into the model
model_svc.fit(x_train, y_train)

# predicting the results for the test set
y_pred = model_svc.predict(x_test)

# calculating the training and testing accuracies
print("Training accuracy :", model_svc.score(x_train, y_train))
print("Testing accuracy :", model_svc.score(x_test, y_test))

#Accuracy Score
print('Accuracy Score of SVC :', accuracy_score(y_test, y_pred))

# classification report
print('classification Report of SVC :\n',classification_report(y_test, y_pred))

# confusion matrix
print('Confusion matrix of SVC :\n',confusion_matrix(y_test, y_pred))
```

```
Training accuracy : 0.9857742683461803
Testing accuracy : 0.9581592580213903
Accuracy Score of SVC : 0.9581592580213903
classification Report of SVC :
              precision    recall  f1-score   support

           0       0.96      1.00      0.98     43111
           1       0.93      0.62      0.75      4761

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

Confusion matrix of SVC :
 [[42896   215]
 [ 1788  2973]]
```

In [49]:
```
from sklearn.model_selection import cross_val_score
CVscore = cross_val_score(model_svc, X, Y, cv =3)
print('Cross Validation Score', model_svc, ':\n')
print("CVScore :" ,CVscore)
print("Mean CV Score :",CVscore.mean())
print("Std deviation :",CVscore.std())
```

```
Cross Validation Score SVC() :

CVScore : [0.95596999 0.95512314 0.95501034]
Mean CV Score : 0.9553678262204993
Std deviation : 0.0004282806702041863
```

Evaluating the model accuracy is an essential part of the process of creating machine learning models to describe how well the model is performing in its predictions are mainly used to evaluate the prediction error rates and model performance in regression analysis.

the best model choose for hyper parameter tuning are Logistic Regression

1)Logistic Regression

Hyper parametertuning

LogisticRegression

In [39]:
```
# Necessary imports
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import GridSearchCV

# Creating the hyperparameter grid
c_space = np.logspace(-5, 8, 15)
param_grid = {'C': c_space}

# Instantiating logistic regression classifier
logreg = LogisticRegression()

# Instantiating the GridSearchCV object
logreg_cv = GridSearchCV(logreg, param_grid, cv = 5)

logreg_cv.fit(X,Y)

# Print the tuned parameters and score
print("Tuned Logistic Regression Parameters: {}".format(logreg_cv.best_params_))
print("Best score is {}".format(logreg_cv.best_score_))
```

```
Tuned Logistic Regression Parameters: {'C': 3.727593720314938}
Best score is 0.9578056104898316
```

In [40]:
```
# creating the model
model = LogisticRegression(C= 3.727593720314938)

# feeding the training set into the model
model.fit(x_train, y_train)

# predicting the results for the test set
y_pred = model_lg.predict(x_test)

# calculating the training and testing accuracies
print("Training accuracy :", model.score(x_train, y_train))
print("Testing accuracy :", model.score(x_test, y_test))

# classification report
print(classification_report(y_test, y_pred))

# confusion matrix
print(confusion_matrix(y_test, y_pred))
```

```
Training accuracy : 0.9711635735324399
Testing accuracy : 0.9598094919786097
              precision    recall  f1-score   support

           0       0.96      1.00      0.98     43111
           1       0.94      0.60      0.73      4761

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

[[42914   197]
 [ 1913  2848]]
```

cross validation score

In [42]:
```
lg_cv=cross_val_score(model,X,Y,scoring='accuracy', cv = 3).mean()
lg_cv
```

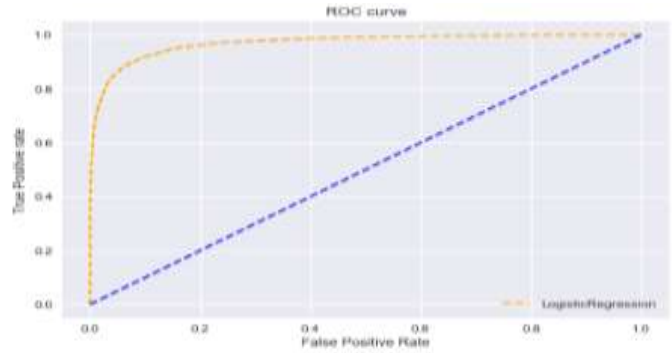
Out[42]: 0.957460939712902

The best model after hyper parameter tuning is LogisticRegression

AUC-ROC curve

```
In [40]: # train model
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier


# RandomForestClassifier
model1 =LogisticRegression()
# DecisionTreeClassifier
model2 = DecisionTreeClassifier()

# fit model
model1.fit(x_train, y_train)
model2.fit(x_train, y_train)

# predict probabilities
pred_prob1 = model1.predict_proba(x_test)
pred_prob2 = model2.predict_proba(x_test)
```

```
In [41]: from sklearn.metrics import roc_curve

# roc curve for models
fpr1, tpr1, thresh1 = roc_curve(y_test, pred_prob1[:,1], pos_label=1)
fpr2, tpr2, thresh2 = roc_curve(y_test, pred_prob2[:,1], pos_label=1)

# roc curve for tpr = fpr
random_probs = [0 for i in range(len(y_test))]
p_fpr, p_tpr, _ = roc_curve(y_test, random_probs, pos_label=1)
```



Saving The Model

```
In [ ]: import joblib
joblib.dump(predRFC,"final model.pkl")
```

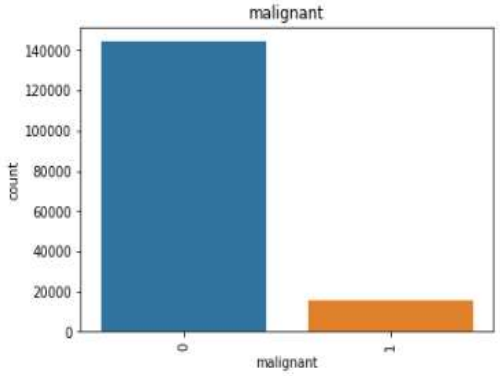
In our project we had implemented various Machine Learning Algorithms such as Logistic Regression, Decision Tree Regression, Random Forest Regression and compared the accuracy of results based on our test data set. Based on the various accuracy levels we find that Logistic Regression gives the highest accuracy i.e. 95%. Therefore we selected Logistic Regression and created User Interface based on it.

Visualizations

After cleaning the data, we can visualize data and better understand the relationships between different variables. There are many more visualizations that you can do to learn more about your dataset, like scatterplots, histograms, boxplots.

- The Analysis of malignant of comments based on data

```
In [12]: sns.countplot(trn_data['malignant'])
         plt.xticks(rotation=90)
         plt.title('malignant')

Out[12]: Text(0.5, 1.0, 'malignant')
```




```
In [13]: sns.countplot(trn_data['rude'])
         plt.xticks(rotation=90)
         plt.title('rude')

Out[13]: Text(0.5, 1.0, 'rude')
```

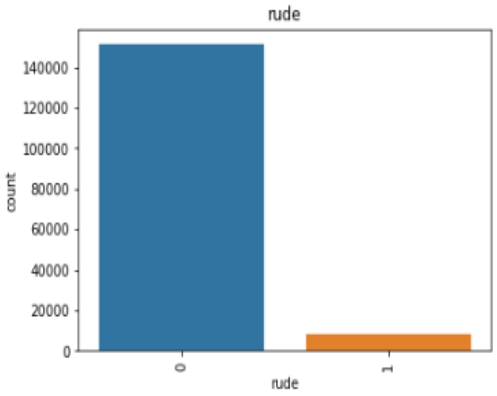


Fig. 1&2: The Analysis of malignant of comments based on data

- While maximum Negative comments categories belong to Malignant, a lot of comments are abusive and rude as well; while threat comments are the minimum.

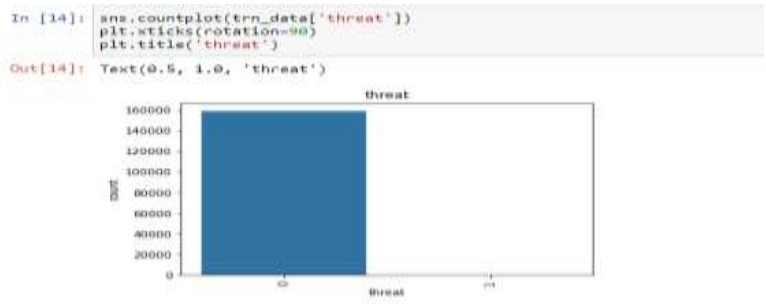


Fig. 3 : The Analysis of Threat comments based on data In

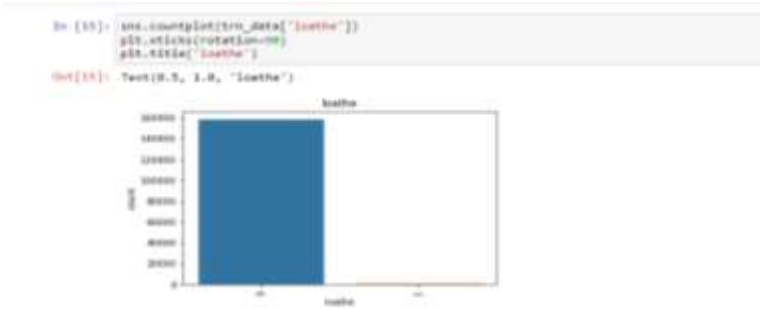


Fig. 4: : The Analysis of Loathe of comments based on data.

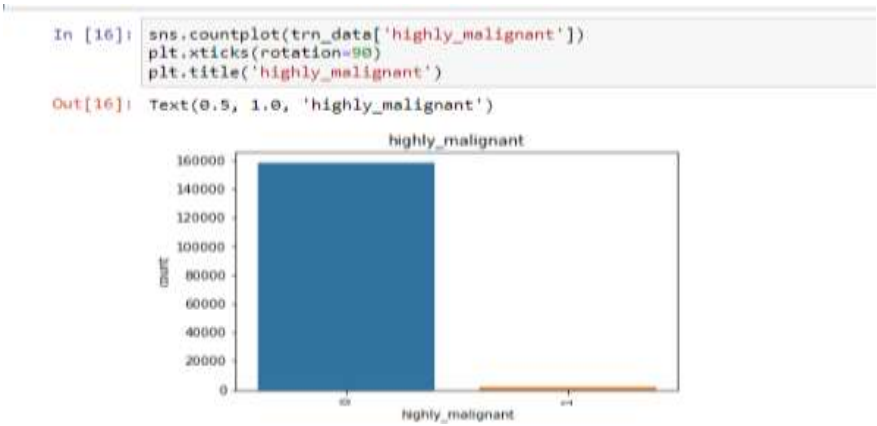
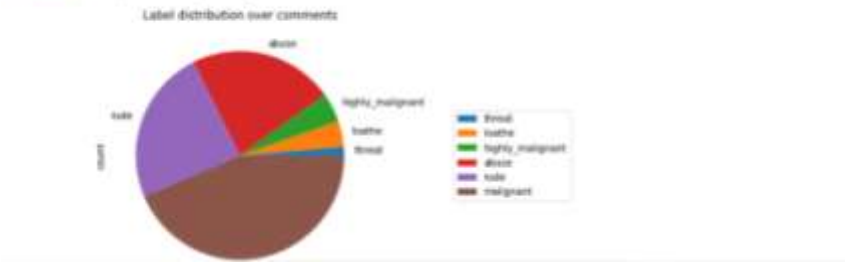


Fig5: The Analysis of highly malignant of comments based on data

```
In [17]: cols_target = ['malignant','highly_malignant','rude','threat','abuse','loathe']
df_distribution = trn_data[cols_target].sum()\
                            .to_frame()\
                            .rename(columns={0: 'count'})\
                            .sort_values('count')

df_distribution.plot.pie(y='count',
                                    title='Label distribution over comments',
                                    figsize=(5, 5))\
                            .legend(loc='center left', bbox_to_anchor=(1.3, 0.5))
```

```
Out[17]: <matplotlib.legend.Legend at 0x2188339dc88>
```



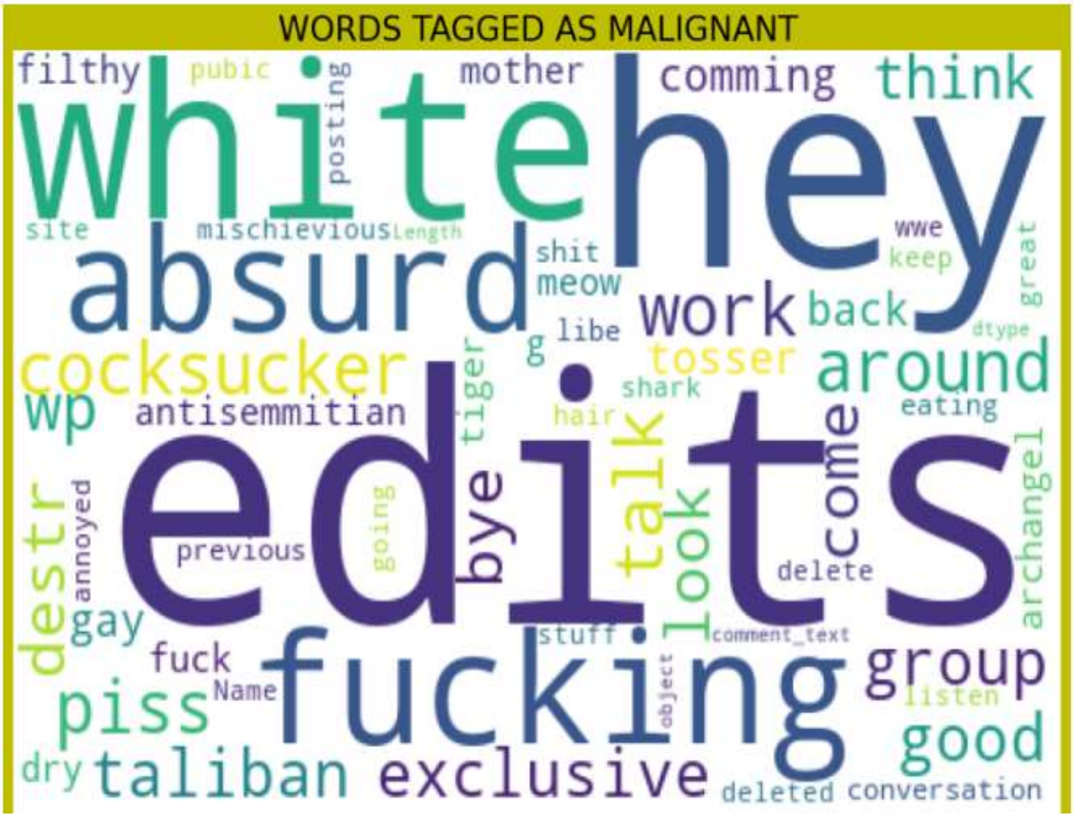
Observation :

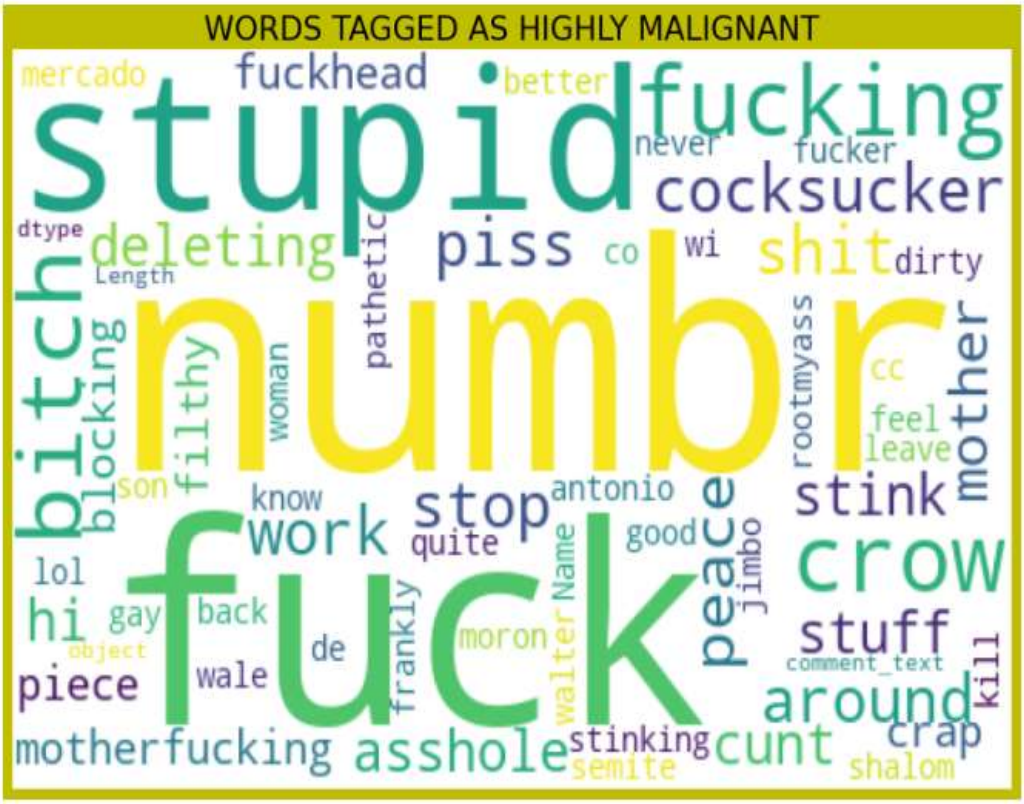
- Around 90% comments are Good/Neutral in nature while rest 10% comments are Negative in nature.
- Out of total negative comments around 43.58% are malignant in nature followed by 24.07% are rude comments.

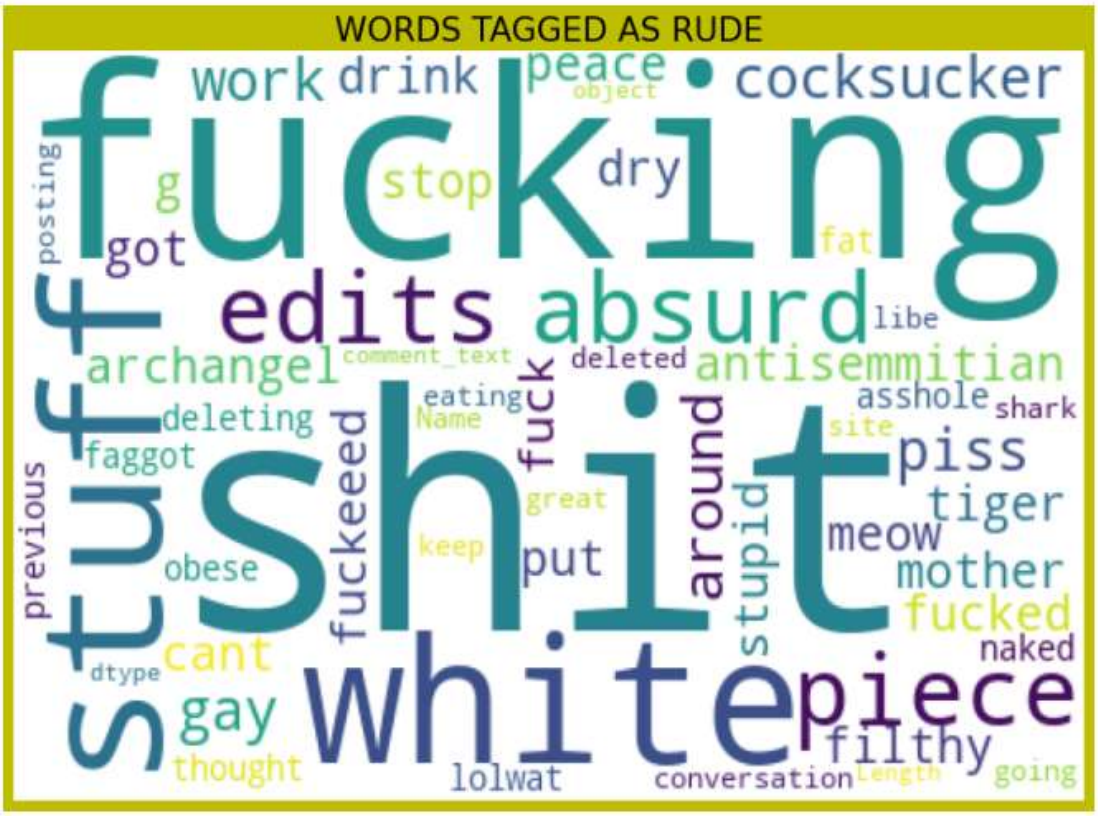
word cloud

➢ A word cloud is a simple yet powerful visual representation object for text processing, which shows the most frequent word with bigger and bolder letters, and with different colors. The smaller the the size of the word the lesser it’s important

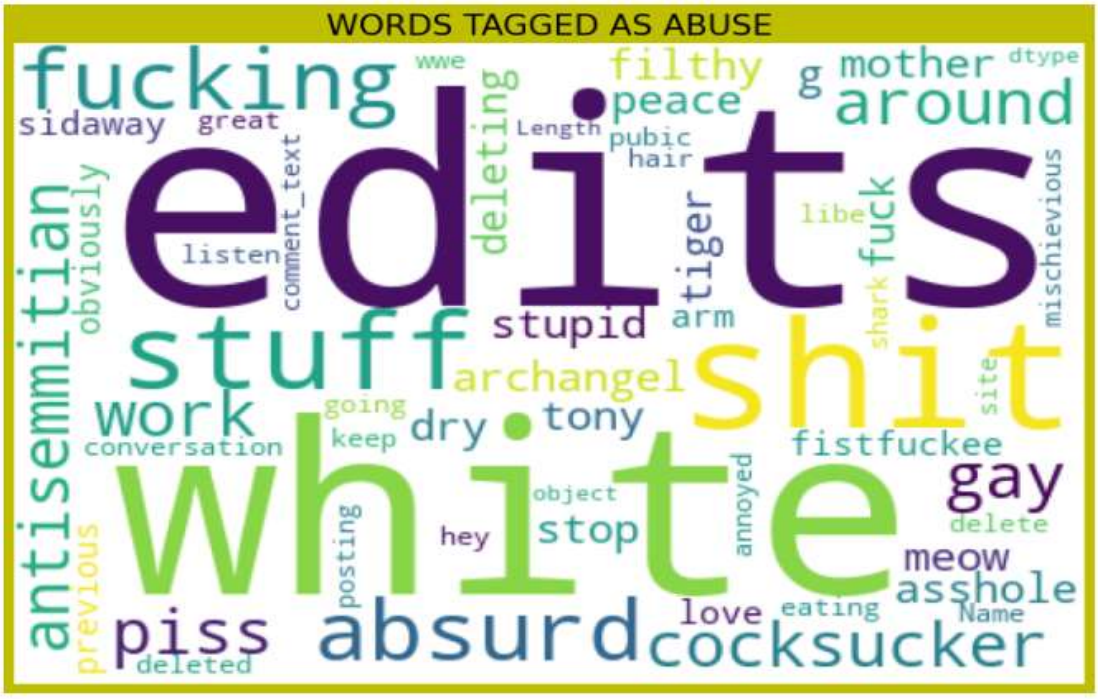
➢ Significant textual data points can be highlighted using a word cloud

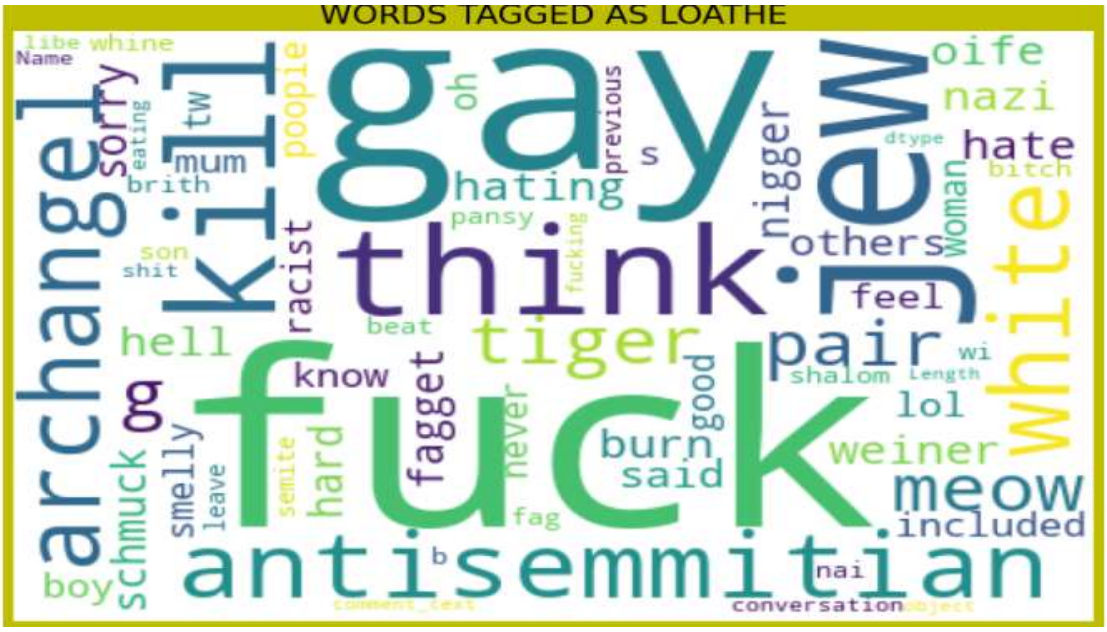

WORDS TAGGED AS MALIGNANT
filthy
pubic
posting
mother
comming
think
white
hey
site
mischievious
Length
shit
wwe
keep
great
absurd
meow
work
back
dtype
libe
g
cocksucker
tosser
around
tiger
shark
wp
antisemmitian
hair
eating
edits
talk
come
archangel
bye
look
destr
annoyed
going
previous
delete
gay
stuff
comment_text
fuck
Name
object
fucking
group
piss
listen
good
dry
taliban
exclusive
deleted
conversation


WORDS TAGGED AS HIGHLY MALIGNANT
mercado
fuckhead
better
fucking
stupid
never
fucker
cocksucker
dtype
deleting
pathetic
piss
co
wi
shit
dirty
Length
numbr
bitch
blocking
filthy
woman
rootmyass
cc
feel
leave
mother
son
know
stop
antonio
stink
work
quite
good
peace
jimbo
lol
crow
hi
gay
back
frankly
Name
moron
walter
stuff
kill
object
de
comment_text
piece
wale
fuck
around
motherfucking
asshole
stinking
cunt
crap
semite
shalom


WORDS TAGGED AS RUDE
fucking
shit
stuff
white
edits
absurd
work
drink
peace
cocksucker
stop
dry
got
archangel
antisemmitian
deleting
faggot
fuckeeed
fuck
around
stupid
piss
tiger
meow
mother
fucked
obese
put
cant
gay
piece
filthy
thought
lolwat
conversation
naked
going
posting
previous
deleted
asshole
shark
fat
libe
eating
Name
site
great
keep
dtype
Length
object
comment_text


WORDS TAGGED AS THREAT
die
suck
bitch
burn
stupid
fagget
hey
back
fuck
hell
undoing
account
stop
hi
think
sorry
hate
last
edits
warning
sodomize
puppet
sock
hole
please
suprise
grave
wife
head
first
smith
find
dance
hore
live
man
notified
sincerely
want
butt
oife
fagg
faggot
dumb
gay
big
dick
time
fucking
watching object
semite
getting
Name
dead
son
yet
Length
musulmans
go
kill
shalom
leave
death
mom
wi
dtype
comment_text





Observation:

- From wordcloud of malignant comments, it is clear that it mostly consists of words like edits, hey, white, fucking, gay, cocksucker, work, think, taliban ect.

- From wordcloud of Highly malignant comments, it is clear that it mostly consists of words like fuck, stupid, fucking, bitch, crow, shit, cocksucker etc.
- From wordcloud of Rude comments, it is clear that it mostly consists of words like fucking, shit, white, piece, edits, stuff, absurd etc.
- From wordcloud of Threat comments, it is clear that it mostly consists of words like fuck,suck, Bitch, die, stupid, etc.
- From wordcloud of Abuse comments, it is clear that it mostly consists of words like edits, white, shit, stuff, fuck, piss, fucking etc.
- From wordcloud of Loathe comments, it is clear that it mostly consists of words like fuck,gay, kill, think, jew, u etc.

CONCLUSION

➢ Key Findings and Conclusions of the Study

From this dataset I get to know that each feature play a very import role to understand the data. Data format plays a very important role in the visualization and Appling the models and algorithms . Logistic Regression performs better with Accuracy Score: 95.59241310160 % than the other classification models. Final Model (Hyper parameter Tuning) is giving us Accuracy score of 96.02% which is slightly improved compare to earlier Accuracy score of 96.02%.Logistic Regression is fastest algorithm compare to others.

There are many systems which uses different machine learning algorithms such as Logistic Regression (LR), Decision Tree, Support Vector Machine (SVM), Random Forest Algorithm, etc for predicting the rating of product. In this ML based system, we are using Logistic Regression Algorithm which gives more accuracy in predicting the rating. It is easy to use and it gives more accuracy in prediction. It requires less time for prediction and it helps in reduction of over fitting. The goal of this dissertation is to successfully predict a user's numerical rating from its review text content. To do so, supervised machine learning techniques and more specifically text classification are used. Three distinct approaches are presented, namely binary classification, aiming at predicting the rating of a review as low or high, as well as multi-class classification and logistic regression whose aim is to predict the exact value of the rating for each review. These datasets are divided into two major categories: experience and search products and are characterized by an imbalanced distribution. We overcome this issue by applying sampling techniques to even out the class distributions. Eventually, the performance of those classifiers is tested and assessed thanks to accuracy metrics, including precision, recall and f1-score.

Learning Outcomes of the Study in respect of DataScience

Learnt how to process the large number of data. Tried and learnt more about distribution of the data. The power of visualization is helpful for the understanding of data into the graphical representation its help me to understand that what data is trying to say, Data cleaning is one of the most important step to remove missing value or null value fill it by mean median or by mode or by 0.Setting a good parameters is more important for the model accuracy. Finding a best random state played a vital roll in finding a better model.

Limitations of this work and Scope for Future Work

The current project predicts the type or toxicity in the comment. We are planning to add the following features in the future:

- Analyse which age group is being toxic towards a particular group or brand.
- Add feature to automatically sensitize words which are classified as toxic.
- Automatically send alerts to the concerned authority if threats are classified as severe.

- Build a feedback loop to further increase the efficiency of the model.
- Handle mistakes and short forms of words to get better accuracy of the result.
- The Maximum feature used while vectorization is 2000. Employing more feature in vectorization lead to more accurate model which I not able to employed due computational resources.
- Data is imbalanced in nature but due to computational limitation we have not employed balancing techniques here.
- Deep learning CNN, ANN can be employed to create more accurate model.