NAME OF THE PROJECT

Malignant Comments Classifier Project

Submitted by:

Mrs. Swati Amit Motugade

FLIPROBO SME:

Ms. Gulshana Chaudhari

ACKNOWLEDGMENT

I would like to express my special gratitude to “Flip Robo” team, who has given me this opportunity to deal with a beautiful dataset and it has helped me to improve my analysis skills. Also, I want to express my huge gratitude to Ms. Gulshana Chaudhari Mam (SME, Flip Robo), she is the person who has helped me to get out of all the difficulties I faced while doing this project.

A huge thanks to “Data trained” who are the reason behind Internship at Fliprobo. Last but not least my parents who are there to support me at every step of my life.

References used in this project:

- SCIKIT Learn Library Documentation.
- Blogs from towardsdatascience, Analytics Vidya, Medium.
- Andrew Ng Notes on Machine Learning (GitHub).
- Data Science Projects with Python Second Edition by Packt
- Hands on Machine learning with scikit learn and tensor flow by Aurelien Geron.
- Research article on Identification and Classification of Toxic Comment Using Machine Learning Methods published in Turkish Journal of Computer and Mathematics Education.
- Article on Toxic Comment Classification by Sara Zaheri, Jeff Leath and David Stroud.

Also, there are so many people who helped me directly and indirectly to complete this project.

Mrs. Swati Amit Motugade

Chapter 1

Introduction

1.1 Business Problem Framing

The increase in penetration of usage of internet services has increased exponentially in the past few years due to the ongoing pandemic, this has empowered an enormous number of dynamic new and old clients utilizing the web for different administrations ranging from academic, entertainment, industrial, monitoring and the emergence of a new trend in the corporate life i.e work-from-home. Due to this sudden emergence of the crowd using the web, there has been an ascent in the number of mischievous persons too. Now it is the primary task of every online platform provider to keep the conversations constructive and inclusive. The best example can be referred to, can be twitter, a web-based media stage where people share their views. This platform has already drawn a lot of flak because of the spread of hate speech, insults, threat, defamatory acts which becomes a challenge for many such online providers in regulating them. Thus, there is active research being conducted in the field of Toxic comment classification. Here we collate non-identical machine learning and other trivial techniques on the dataset and propose a model that outflanks all others and compares them one-on-one. And the results would help up to create an online interface where we would be able to identify the toxicity level in the given phrase or sentence and classify them into their order of toxicity.

1.2 Conceptual Background of the Domain Problem

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 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 unoffensive, 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 cyberbullying.

Data Set Description

The data set contains the training set, which has approximately 1,59,000 samples and the test set which contains nearly 1,53,000 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.

This project is more about exploration, feature engineering and classification that can be done on this data. Since the data set is huge and includes many categories of comments, we can do good amount of data exploration and derive some interesting features using the comments text column available.

We need to build a model that can differentiate between comments and its categories.

Refer to the data set file provided along with this.

1.3 Review of Literature

Due the penetration of the internet in all domains of life which has led to increase of people's participation actively and give remarks as an issue of communicating their concern/feedback/opinion in various online forums. Although most of the times these comments are helpful for the creator to extemporize the substance that is being provided to people, but sometimes these may be abusive and create hatred-feeling among the people. Thus, as these are openly available to the public which is being viewed from various sections of the society, people in different age groups, different communities and different socio-economic background, it becomes the prime people. responsibility of the content-creator (the host) to filter out these comments in order to stop the spread of negativity or hatred within people.

Lately there has been many cases in which the growing menace of hate and negativity has been witnessed in the online platforms especially social media as such, many governments around the world have seen the rise of cases related to cyber bullying that has led to spread of hatred and violence.

Since the democratization of substance creation following the dispatch of web-based media stages, every single one of us has become content makers making and distributing our own substance, which thus has made a framework where the nature of distributed substance cannot, at this point be controlled. The effect of the most recent twenty years' innovation unrest is presently affecting organizations, political frameworks, family lives, society, and individuals.

Detecting Toxic comments has been a great challenge for the all the scholars in the field of research and development. This domain has drawn lot of interests not just because of the spread of hate but also people refraining people from participating in online forums which

diversely affects for all the creators/content-providers to provide a relief to engage in a healthy public interaction which can be accessed by public without any hesitation.

There have been sure turns of developments in this area which includes couple of models served through API. But the models still make errors and still fail to provide an accurate solution to the problem. In this paper we have widely discussed a set of models which is utilized for text classification.

There is a difference between the traditional and very famous multi-class classification, and the one which we will be using, which is the multi-label classification. In a multi-class classification, each instance is classified into one of three or more classes, whereas, in a multi-label classification, multiple labels (such as – toxic, severe-toxic, obscene, threat, insult or identityhate) are to be predicted for the same instance.

Multiple ways are there to approach this classification problem. It can be done using –

→ Multi-label methods which belong to the problem transformation category: Label Power Set (LP), Binary Relevance (BR), BR+ (BRplus), and classifier chain.

→ Base and adapted algorithms like: J48 (Decision Tree), Naïve Bayes, k-Nearest-Neighbor (KNN), SMO (Support Vector Machines), and, BP-MLL neural networks.

Further, out of the total dataset used for experimenting these algorithms, 70% was used for training and 30% was used for testing. Each testing dataset was labelled and thus for each algorithm using the predictions and labels, calculation of metric such as hamming-loss, accuracy and log-loss was done. The final results have been complied on the basis of values obtained by algorithmic models in hamming-loss and log-loss combined.

Evaluation Metrics

→ Label bases metrics include one-error, average precision, etc. These can be calculated for each label, and then can be averaged for all without taking into account any relation between the labels if exists.

Average Precision (AP): Average precision is a measure that combines recall and precision for ranked retrieval results. For one information need, he averages precision is the mean of the precision scores after each relevant document is retrieved, where, and are the precision and recall at the threshold.

→ Example based metrics include accuracy, hamming loss, etc. These are calculated for each of the examples and then averaged across the test set. Let –

Accuracy is defined as the proportion of correctly predicted labels to the total no. of labels for each instance.

Hamming-loss is defined as the symmetric difference between predicted and true labels, divided by the total no. of labels.

When we have a look at the data, we observe that every 1 in 10 samples is toxic, every 1 in 50 samples is obscene and insulting, but the occurrences of sample being severe toxic, threat and identity hate is extremely rare. Thus, we have skewed data, and accuracy as metric will not give us the required results. Thus, we will be using hamming-loss as the evaluation metric.

1.4 Motivation for the Problem Undertaken

The project was the first provided to me by Flip Robo Technologies as a part of the internship programme. The opportunity to deploy my skillset in solving real time problem has been the primary motivation.

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.

Chapter 2

Analytical Problem Framing

2.1 Dataset Description

The dataset consists of the following fields-

- **id: An 8-digit integer value, to get the identity of the person who had written this comment**
- **comment_text: A multi-line text field which contains the unfiltered comment**
- **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.**

Out of these fields, the comment_text field will be pre-processed and fitted into different classifiers to predict whether it belongs to one or more of the labels/outcome variables (i.e., Malignant, Highly Malignant, Rude, Threat, Abuse, Loathe)
We have a total of 159571 samples of comments and labelled data, which can be loaded from train.csv file. The first 5 samples are as follows:

t[3]:

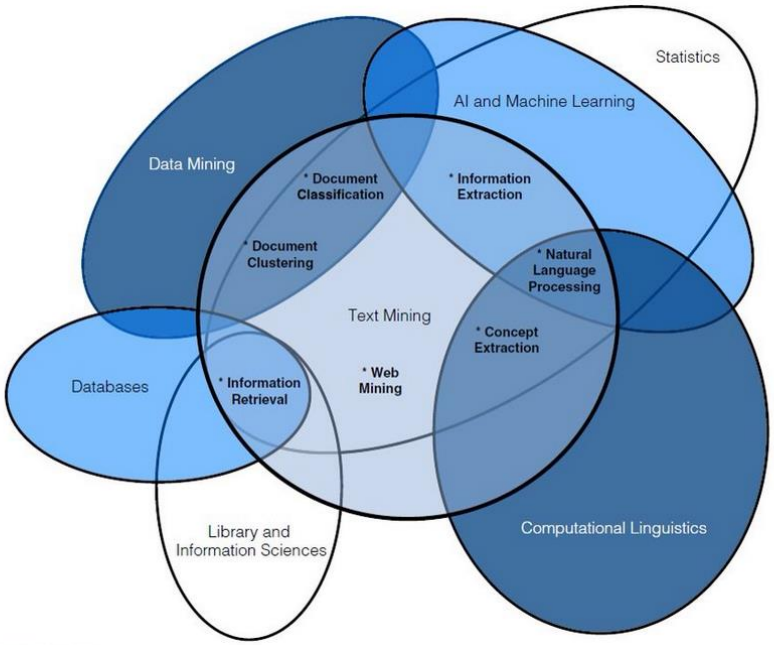
	id	comment_text
0	0000997932d777bf	Explanation\nWhy the edits made under my usern...
1	000103f0d9cfb60f	D'aww! He matches this background colour I'm s...
2	000113f07ec002fd	Hey man, I'm really not trying to edit war. It...
3	0001b41b1c6bb37e	"\nMore\nI can't make any real suggestions on ...
4	0001d958c54c6e35	You, sir, are my hero. Any chance you remember...

One more very crucial aspect of the dataset is noticing the frequency occurrence of multilabel data. We can easily notice that approximately every 1 data out of 10 is Malignant, every 1 in 20 samples is Rude and insulting.

2.2 Text Mining

It's the process of extracting non-trivial, high quality and interesting info from unstructured text. Its framework is similar to ETL (Extract, Transform, Load)
Text mining process starts with pre-processing. That transforms input raw text in structured information. Text pre-processing input is unstructured data or semi structured data such as HTML pages. The data submitted to this process is cleaned and useful features are recovered. The output is stored in database or any other structured format. Text Mining techniques are applied to this structured form of data. The different data

mining algorithms can be applicable to this data to model text data with applications such as information retrieval system.



1) Punctuation Marks:

Removing all punctuation marks. There are different libraries used for removing punctuation marks.

2) Numbers:

Remove all the numbers or transform them into words from the text data that includes timeline, dates, ip addresses, etc.

3) Case folding:

Converting all letters to lower-case.

4) Stop Words (removing unnecessary words):

It generally includes removing high frequency words such as (a, an, the, all pronouns, etc). This step is highly depended on language. Stop words are stored in hash table according to the domain knowledge. Stop words are predefined in for English language in most of the libraries. But if any word is found not giving important information can also be labelled as stop word manually along with the predefined ones. There is no universal

stopword list, but a standard English language stopwords list from nltk library can be used. Also, domain-specific stopwords can be added.5)

5) White spaces:

Removing white spaces that may be present at the start or end of the sentences/words or there can be extra spaces at anywhere in the sentence.

6) Stemming:

Reducing the tokens to root forms to recognize morphological variations. It blindly strips off any prefixes or postfixes in iterative manner.

7) Lemmatization:

Converting variant forms to the base forms. This directly impacts vocabulary size. This prevents duplication of data by linking words to root word. For example, “am” , “are” are linked to “be”. To achieve this, we need list grammatical rules and a list of irregular words. Practically, use Wordnet’s “morphstr” function.

8) Synonym check:

Words with the same form and multiple related meanings (e.g., bank: a financial institution and bank: to rely upon, as in "You can bank on me") can't be treated as one entity. Words with same meaning can only be treated as one entity.

9) Tokenization:

The process of breaking down a text paragraph into smaller chunks such as words or sentence is called Tokenization. Token is a single entity that is building blocks for sentence or paragraph. It's nothing but converting text into tokens. Splitting whole corpus into smaller parts (usually a word). (e.g., Bag of

words model, N-Gram model). Tokenizing can be simply achieved by splitting the text into white spaces.

2.3 Data Loading & Integrity Check

Both the training and testing datasets are provided in the form of csv files. We have to load them into jupyter notebook by using pandas' function to load csv files.

Loading Training dataset

```
df_train = pd.read_csv(r"C:\Users\Swati\OneDrive\Desktop\DataScience\Projects\Malignant Comments Classifier Project\train.cs
df_train.head()
```

	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	"\nMore\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

Loading Test dataset

```
df_test = pd.read_csv(r"C:\Users\Swati\OneDrive\Desktop\DataScience\Projects\Malignant Comments Classifier Project\test.csv"
df_test.head()
```

	id	comment_text
0	00001cee341fdb12	Yo bitch Ja Rule is more succesful then you'll...
1	0000247867823ef7	== From RfC == \n\n The title is fine as it is...
2	00013b17ad220c46	" \n\n == Sources == \n\n * Zawe Ashton on Lap...
3	00017563c3f7919a	:If you have a look back at the source, the in...
4	00017695ad8997eb	I don't anonymously edit articles at all.

Checking for shape of both datasets

```
print('No. of Rows :',df_train.shape[0])
print('No. of Columns :',df_train.shape[1])
pd.set_option('display.max_columns',None) # This will enable us to see truncated columns
df_train.head()
```

```
No. of Rows : 159571
No. of Columns : 8
```

```
print('No. of Rows :',df_test.shape[0])
print('No. of Columns :',df_test.shape[1])
pd.set_option('display.max_columns',None) # This will enable us to see truncated columns
df_test.head()
```

```
No. of Rows : 153164
No. of Columns : 2
```

The training dataset has 159574 rows and 8 columns whereas the test dataset has 153164 rows and 2 columns.

Check for Missing values

```
1 df_train.isnull().sum().sum()
```

```
0
```

```
1 df_test.isnull().sum().sum()
```

```
0
```

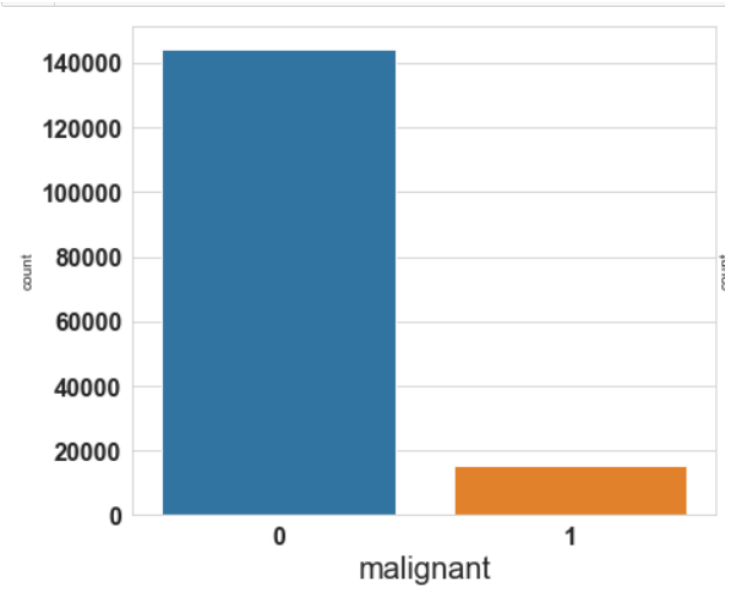
Both the datasets have no missing values.

Chapter 3

Exploratory Data Analysis

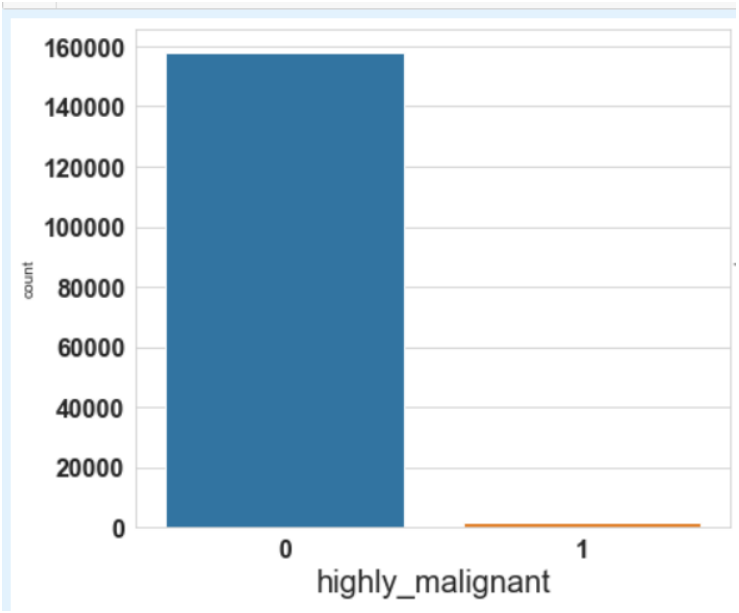
EDA or Data Visualization is the process of analysing data in the form of graphs or maps, making it a lot easier to understand the trends or patterns in the data.

1. Malignant



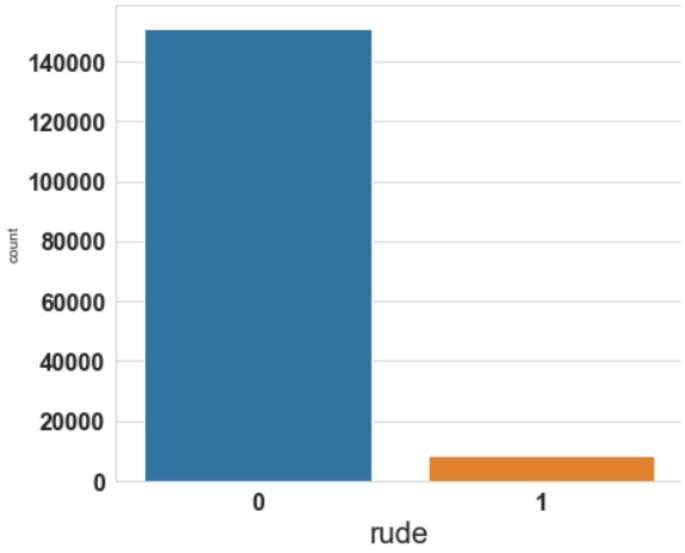
Out of all comments, 15294 comments are malignant.

2. Highly Malignant



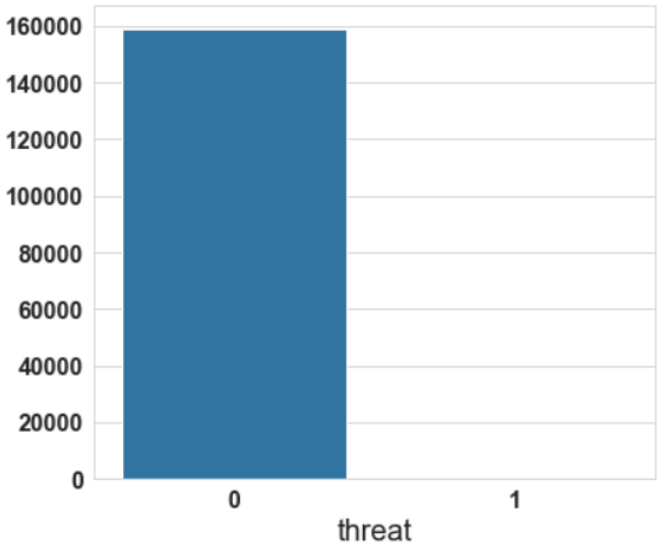
Out of all comments, 1595 comments are Highly malignant.

3. Rude



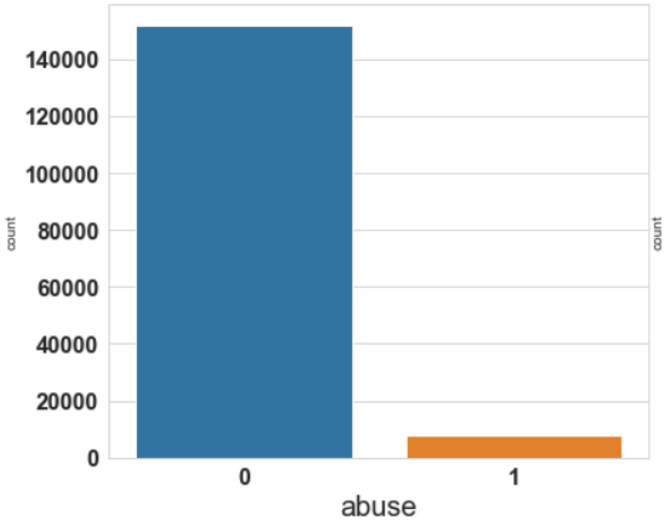
Out of all comments, 8449 comments are Rude.

4. Threat



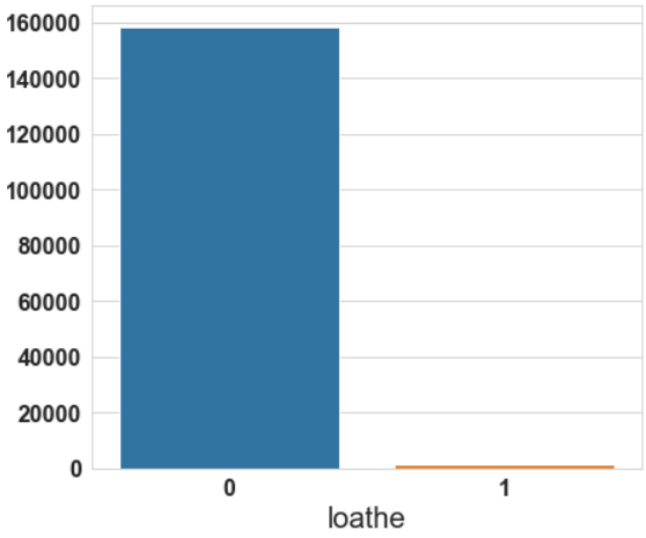
Out of all comments, 478 comments are Rude.

5. Abuse



Out of all comments, 7877 comments are Rude.

6. Loathe



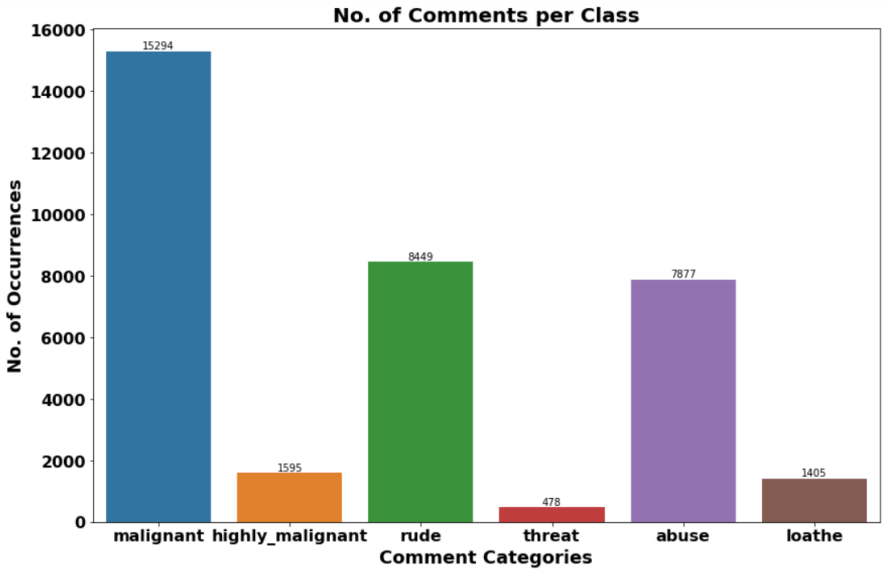
Out of all comments, 1405 comments are Rude.

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

Around 90% comments are Good/Neutral in nature while rest 10% comments are Negative in nature.

Distribution of Comments

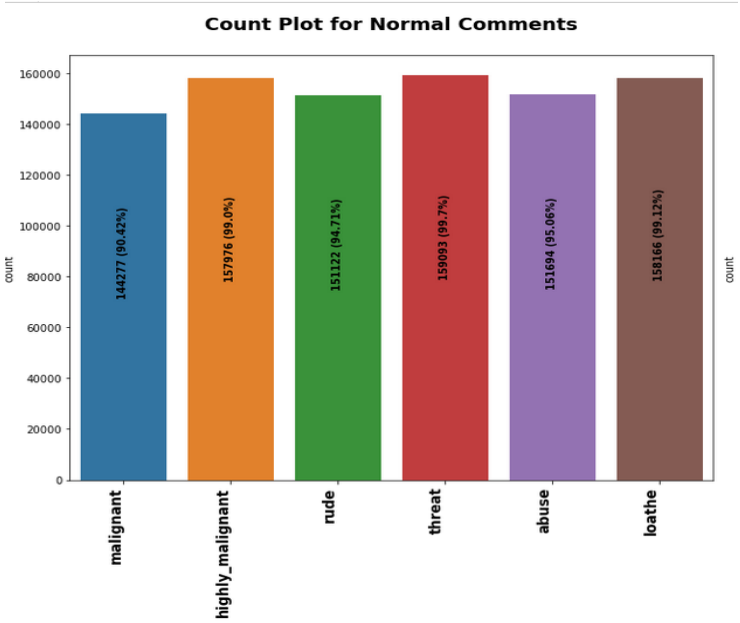
Number of comments per class

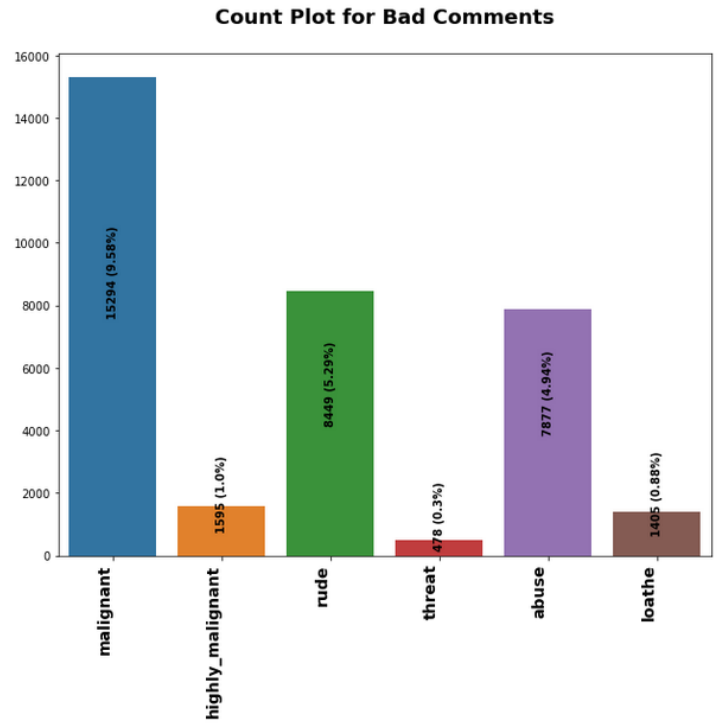


1. Out of total Negative comments the maximum negative comments come with Malignant in nature followed by rude categories.

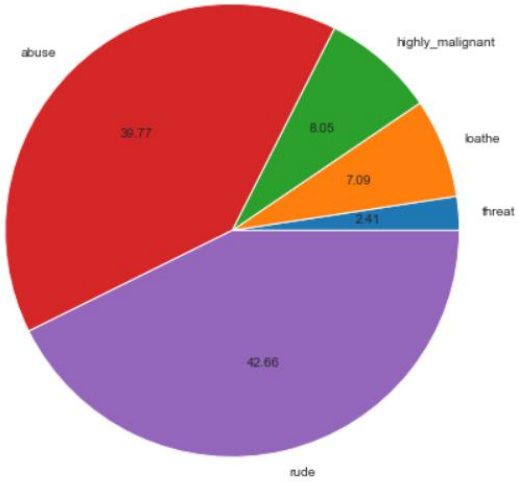
2. Very few comments come with threatening nature.

Count plot for Normal & Bad comments



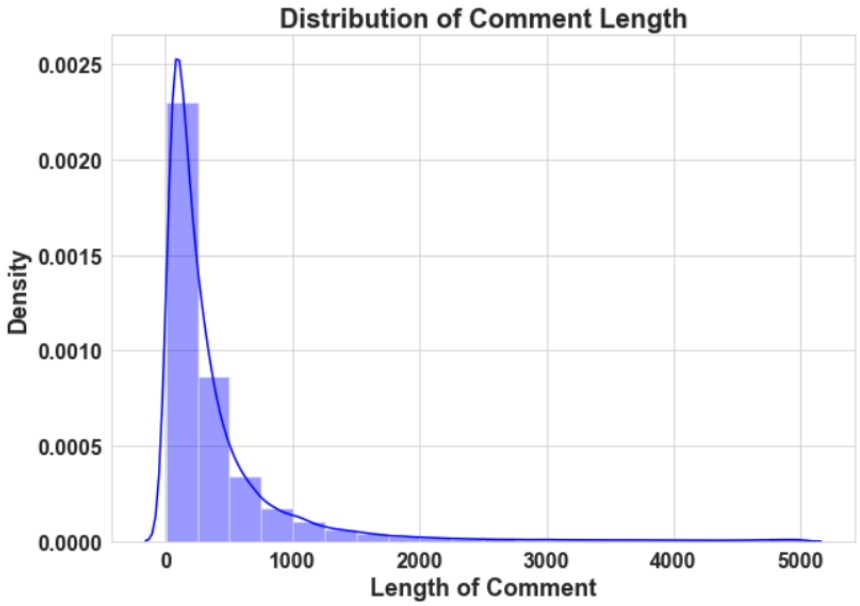


Label Distribution over comments



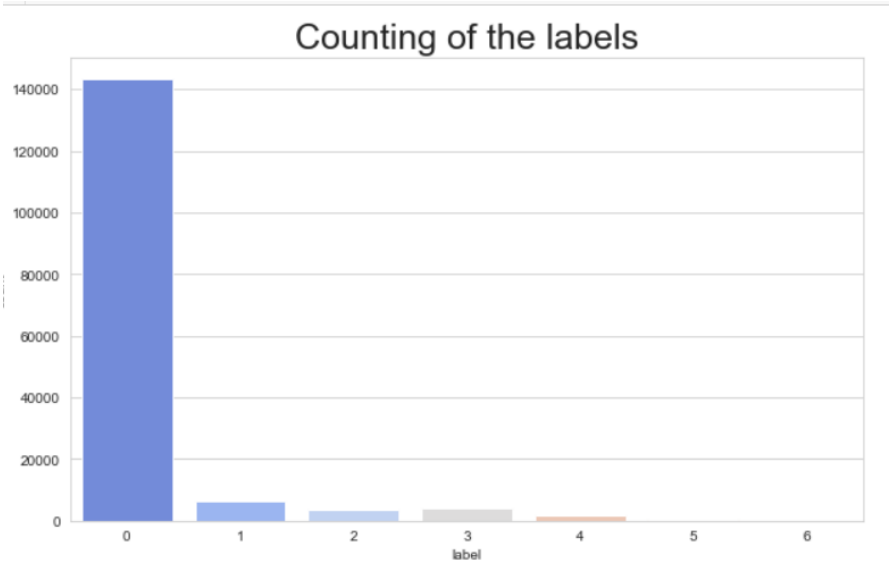
1. Around 90% comments are Good/Neutral in nature while rest 10% comments are Negative in nature.

2. Out of total negative comments around 43.58% are malignant in nature followed by 24.07% are rude comments.

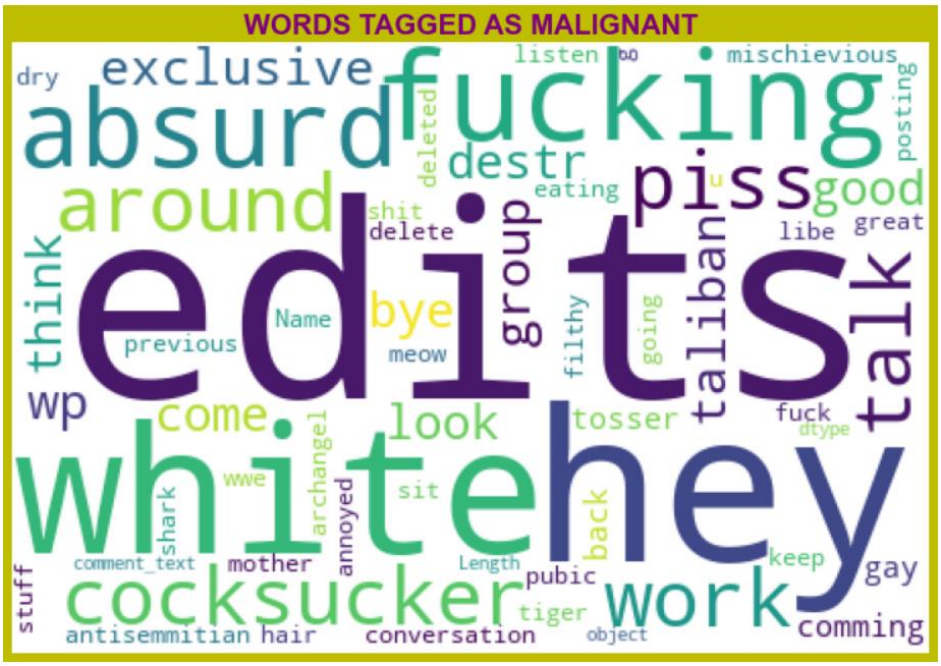


Above is a plot showing the comment length frequency. As noticed, most of the comments are short with only a few comments longer than 1000 words. Majority of the comments

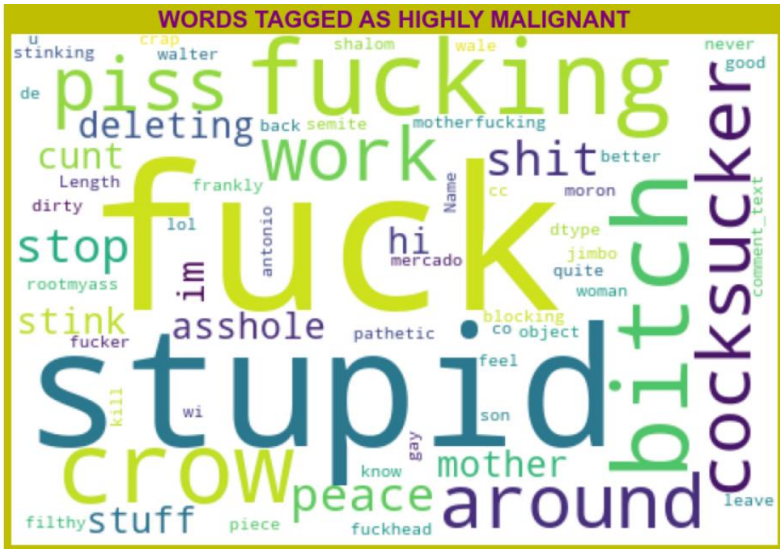
are of length 500, where maximum length is 5000 and minimum length is 5. Median length being 250.



Word Cloud for WORDS TAGGED AS Bad Comments

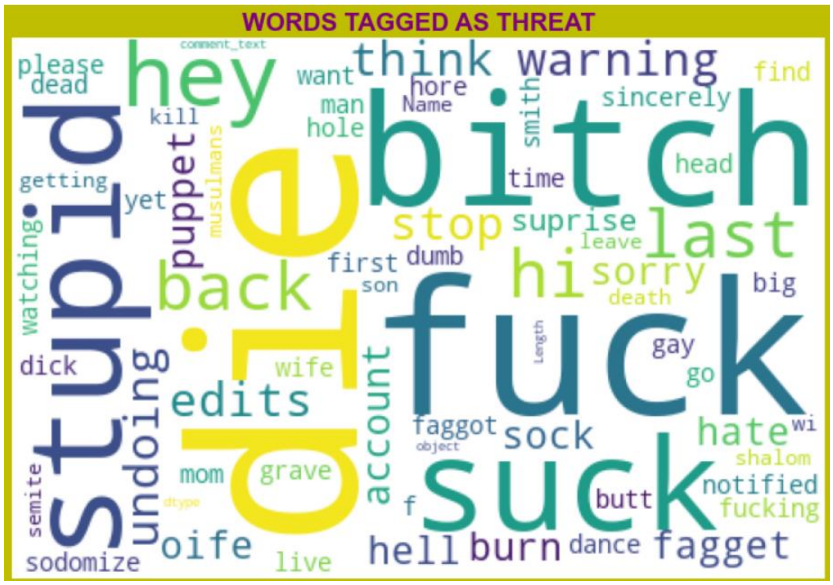


From wordcloud of malignant comments, it is clear that it mostly consists of words like edits, hey, white, fucking, gay, cocksucker, work, think, taliban etc.



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.

Chapter 4

Feature Engineering

4.1 Correlation

```
#Checking correlation of the dataset
corr=df_train.corr()
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

1.The highest positive correlation is seen in between fields 'rude' and 'abuse'.

4.2 Removing Unnecessary Variables

```
#As ID is not much important, we can drop from the dataset
df_train.drop('id',axis=1,inplace=True)
df_test.drop('id',axis=1,inplace=True)
```

4.3 Feature Extraction for length before cleaning

```
# Creating a column 'length_before_cleaning' in training dataset
# It represents the length of the each comment respectively in a column 'comment_text'
df_train['length_before_cleaning'] = df_train['comment_text'].map(lambda comment_text: len(comment_text))
df_train
```

```
# Creating a column 'length_before_cleaning' in test dataset
# It represents the length of the each comment respectively in a column 'comment_text'
df_test['length_before_cleaning'] = df_test['comment_text'].map(lambda comment_text: len(comment_text))
df_test.head(5)
```

4.4 Data Mining

```
#Importing Required libraries
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
```

```
#Defining the stop words
stop_words = stopwords.words('english')

#Defining the lemmatizer
lemmatizer = WordNetLemmatizer()
```

```
#Replacing '\n' in comment_text
df_train['comment_text'] = df_train['comment_text'].replace('\n',' ')
```

text preprocessing for getting cleaned texts

```python
#Function Definition for using regex operations and other text preprocessing for getting cleaned texts
def clean_comments(text):

    #convert to lower case
    lowered_text = text.lower()

    #Replacing email addresses with 'emailaddress'
    text = re.sub(r'^.+@[^\.].*\.[a-z]{2,}$', 'emailaddress', lowered_text)

    #Replace URLs with 'webaddress'
    text = re.sub(r'http\S+', 'webaddress', text)

    #Removing numbers
    text = re.sub(r'[0-9]', " ", text)

    #Removing the HTML tags
    text = re.sub(r"<.*?>", " ", text)

    #Removing Punctuations
    text = re.sub(r'[^\w\s]', ' ', text)
    text = re.sub(r'\_',' ',text)

    #Removing all the non-ascii characters
    clean_words = re.sub(r'[^\x00-\x7f]',r'', text)

    #Removing the unwanted white spaces
    text = " ".join(text.split())

    #Splitting data into words
    tokenized_text = word_tokenize(text)

    #Removing remaining tokens that are not alphabetic, Removing stop words and Lemmatizing the text
    removed_stop_text = [lemmatizer.lemmatize(word) for word in tokenized_text if word not in stop_words if word.isalpha()]

    return " ".join(removed_stop_text)
```

Replace original comment with cleaned text in training dataset

```python
# Calling the above function for the column comment_text in training dataset to replace original with cleaned text
df_train['comment_text'] = df_train['comment_text'].apply(clean_comments)
df_train['comment_text'].head()
```

```
0    explanation edits made username hardcore metal...
1    aww match background colour seemingly stuck th...
2    hey man really trying edit war guy constantly ...
3    make real suggestion improvement wondered sect...
4                      sir hero chance remember page
Name: comment_text, dtype: object
```

Creating a column 'len_after_cleaning'

```python
# Creating a column 'len_after_cleaning'
# Representing the length of the each comment respectively in a column 'comment_text' after cleaning the text.
df_train['length_after_cleaning'] = df_train['comment_text'].map(lambda comment_text: len(comment_text))
df_train.head()
```

	comment_text	malignant	highly_malignant	rude	threat	abuse	loathe	label	length_before_cleaning	length_after_cleaning
0	explanation edits made username hardcore metal...	0	0	0	0	0	0	0	264	156
1	aww match background colour seemingly stuck th...	0	0	0	0	0	0	0	112	67
2	hey man really trying edit war guy constantly ...	0	0	0	0	0	0	0	233	141
3	make real suggestion improvement wondered sect...	0	0	0	0	0	0	0	622	364
4	sir hero chance remember page	0	0	0	0	0	0	0	67	29

Checking Total length removal in train dataset

```python
# Checking Total length removal in train dataset
print("Original Length:", df_train.length_before_cleaning.sum())
print("Cleaned Length:", df_train.length_after_cleaning.sum())
print("Total Words Removed:", (df_train.length_before_cleaning.sum()) - (df_train.length_after_cleaning.sum()))
```

```
Original Length: 62893130
Cleaned Length: 38474840
Total Words Removed: 24418290
```

Replace original comments with cleaned text in test dataset

```
# Calling the above function for the column comment_text in test dataset so that we can replace original with cleaned text
df_test['comment_text'] = df_test['comment_text'].apply(clean_comments)
df_test['comment_text'].head()
```

```
0    yo bitch ja rule succesful ever whats hating s...
1                                  rfc title fine imo
2                             source zawe ashton lapland
3    look back source information updated correct f...
4                               anonymously edit article
Name: comment_text, dtype: object
```

Creating a column 'len_after_cleaning'

```
#Creating a column 'len_after_cleaning'
#It represents the length of the each comment respectively in a column 'comment_text' after cleaning the text
df_test['length_after_cleaning'] = df_test['comment_text'].map(lambda comment_text: len(comment_text))
df_test.head()
```

	comment_text	length_before_cleaning	length_after_cleaning
0	yo bitch ja rule succesful ever whats hating s...	367	235
1	rfc title fine imo	50	18
2	source zawe ashton lapland	54	26
3	look back source information updated correct f...	205	109
4	anonymously edit article	41	24

Total length removal in test dataset

```
# Total length removal in test dataset
print('Original Length:',df_test.length_before_cleaning.sum())
print('Clean Length:',df_test.length_after_cleaning.sum())
print("Total Words Removed:", (df_test.length_before_cleaning.sum()) - (df_test.length_after_cleaning.sum()))
```

```
Original Length: 55885733
Clean Length: 34282033
Total Words Removed: 21603700
```

Vectorizer & Splitting Train dataset

Converting the features into number vectors

```
# Converting the features into number vectors
tf_vec = TfidfVectorizer(max_features = 2000, stop_words='english')
```

```
# Let's Separate the input and output variables represented by X and y respectively in train data and convert them
X = tf_vec.fit_transform(df_train['comment_text']).toarray()
```

```
output_labels= df_train.columns[1:7]
```

```
# output variables
from scipy.sparse import csr_matrix
Y = csr_matrix(df_train[output_labels]).toarray()

# checking shapes of input and output variables to take care of data imbalance issue
print("Input Variable Shape:", X.shape)
print("Output Variable Shape:", Y.shape)
```

Vectorizer & Splitting Test dataset

```
# Doing the above process for test data
test_vec = tf_vec.fit_transform(df_test['comment_text'])
test_vec
```

```
<153164x2000 sparse matrix of type '<class 'numpy.float64'>'
        with 2138199 stored elements in Compressed Sparse Row format>
```

```
test_vec.shape
```

```
(153164, 2000)
```

Chapter 5

Machine Learning Model Building

5.1 Importing Necessary Libraries

```
!pip install scikit-multilearn
```

```
Requirement already satisfied: scikit-multilearn in c:\users\swati\anaconda3\lib\site-packages (0.2.0)
```

```
#Importing Machine learning Model library
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import MultinomialNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import GradientBoostingClassifier
from xgboost import XGBClassifier
from skmultilearn.problem_transform import BinaryRelevance
from sklearn.svm import SVC, LinearSVC
from sklearn.multiclass import OneVsRestClassifier
from sklearn.model_selection import train_test_split,cross_val_score
from sklearn.metrics import confusion_matrix,classification_report,accuracy_score
from sklearn.metrics import roc_auc_score, roc_curve, auc
from sklearn.metrics import hamming_loss, log_loss
```

```
import timeit, sys
import tqdm.notebook as tqdm
```

5.2 Training and Testing Model on our train dataset

```
def build_models(models,x,y,test_size=0.33,random_state=42):
    # spliting train test data using train_test_split
    x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=test_size,random_state=random_state)

    # training models using BinaryRelevance of problem transform
    for i in tqdm.tqdm(models,desc="Building Models"):
        start_time = timeit.default_timer()

        sys.stdout.write("\n=========================================================================================\n")
        sys.stdout.write(f"Current Model in Progress: {i} ")
        sys.stdout.write("\n=========================================================================================\n")

        br_clf = BinaryRelevance(classifier=models[i]["name"],require_dense=[True,True])
        print("Training: ",br_clf)
        br_clf.fit(x_train,y_train)

        print("Testing: ")
        predict_y = br_clf.predict(x_test)

        ham_loss = hamming_loss(y_test,predict_y)
        sys.stdout.write(f"\n\tHamming Loss  : {ham_loss}")

        ac_score = accuracy_score(y_test,predict_y)
        sys.stdout.write(f"\n\tAccuracy Score: {ac_score}")

        cl_report = classification_report(y_test,predict_y)
        sys.stdout.write(f"\n{cl_report}")

        end_time = timeit.default_timer()
        sys.stdout.write(f"Completed in [{end_time-start_time} sec.]")

        models[i]["trained"] = br_clf
        models[i]["hamming_loss"] = ham_loss
        models[i]["accuracy_score"] = ac_score
        models[i]["classification_report"] = cl_report
        models[i]["predict_y"] = predict_y

        models[i]["time_taken"] = end_time - start_time

        sys.stdout.write("\n=========================================================================================\n")

    models["x_train"] = x_train
    models["y_train"] = y_train
    models["x_test"] = x_test
    models["y_test"] = y_test

    return models
```

5.3 Preparing the list of models for classification purpose

```
# Preparing the list of models for classification purpose
models = {
          "Logistic Regression": {"name": LogisticRegression()},
          "Random Forest Classifier": {"name": RandomForestClassifier()},
          "Support Vector Classifier": {"name": LinearSVC(max_iter = 3000)},
          "Ada Boost Classifier": {"name": AdaBoostClassifier()},
         }

# Taking one forth of the total data for training and testing purpose
half = len(df_train)//4
trained_models = build_models(models,X[:half,:],Y[:half,:])
```

5.4 Preparing the list of models for classification purpose

```
# Preparing the list of models for classification purpose
models = {
          "Logistic Regression": {"name": LogisticRegression()},
          "Random Forest Classifier": {"name": RandomForestClassifier()},
          "Support Vector Classifier": {"name": LinearSVC(max_iter = 3000)},
          "Ada Boost Classifier": {"name": AdaBoostClassifier()},
         }

# Taking one forth of the total data for training and testing purpose
half = len(df_train)//4
trained_models = build_models(models,X[:half,:],Y[:half,:])
```

5.5 Hamming Loss & Accuracy Scores for Loaded Models

Algorithms	Hamming Loss	Accuracy Score
Logistic Regression	0.022066084	0.912343334
Random Forest Classifier	0.021977465	0.907330041
Support Vector Classifier	0.020952019	0.911507785
Ada Boost Classifier	0.023446005	0.905734903

From the above model comparison, it is clear that Linear Support Vector Classifier performs better with Accuracy Score: 91.15077857956704 % and Hamming Loss: 2.0952019242942144 % than the other classification models.

Therefore, I am now going to use Linear Support Vector Classifier for further Hyperparameter tuning process.

5.6 Hyperparameter Tuning

```
from sklearn.model_selection import  GridSearchCV
```

```
fmod_param = {'estimator__penalty' : ['l1', 'l2'],
              'estimator__loss' : ['hinge', 'squared_hinge'],
              'estimator__multi_class' : ['ovr', 'crammer_singer'],
              'estimator__random_state' : [42, 72, 111] }
#SVC = BinaryRelevance(classifier=LinearSVC(),require_dense=[True,True])
SVC = OneVsRestClassifier(LinearSVC())
GSCV = GridSearchCV(SVC, fmod_param, cv=3,verbose = 10)
x_train,x_test,y_train,y_test = train_test_split(X[:half,:], Y[:half,:], test_size=0.30, random_state=42)
GSCV.fit(x_train,y_train)
GSCV.best_params_
```

```
{'estimator__loss': 'hinge',
 'estimator__multi_class': 'ovr',
 'estimator__penalty': 'l2',
 'estimator__random_state': 42}
```

5.7 Final Model

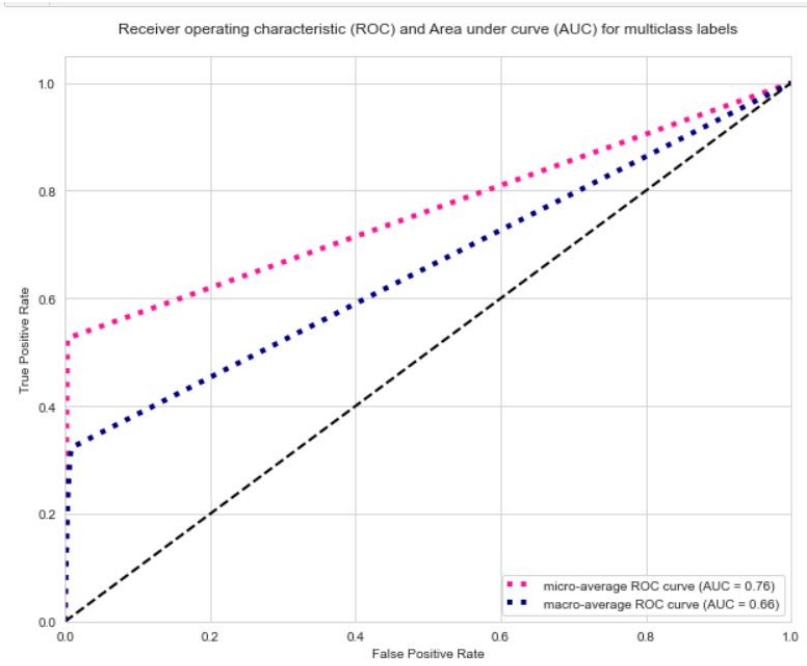
```
Final_Model = OneVsRestClassifier(LinearSVC(loss='hinge',
            multi_class='ovr', penalty='l2', random_state=42))

Classifier = Final_Model.fit(x_train, y_train)
fmod_pred = Final_Model.predict(x_test)
fmod_acc = (accuracy_score(y_test, fmod_pred))*100
print("Accuracy score for the Best Model is:", fmod_acc)
h_loss = hamming_loss(y_test,fmod_pred)*100
print("Hamming loss for the Best Model is:", h_loss)
```

Accuracy score for the Best Model is: 91.26002673796792
Hamming loss for the Best Model is: 2.0819407308377897

Final Model is giving us Accuracy score of 91.26% which is slightly improved compare to earlier Accuracy score of 91.15%.

5.8 AOC - ROC Curve of Final Model



5.9 Confusion Matrix for Final Model

```
1 from sklearn.metrics import roc_curve, auc, roc_auc_score, multilabel_confusion_matrix
2 print("Confusion matrix:\n\n", multilabel_confusion_matrix(y_test, fmod_pred))
```

```
Confusion matrix:

 [[[10720    73]
  [  507   668]]

 [[11833     0]
  [  135     0]]

 [[11268    42]
  [  231   427]]

 [[11930     0]
  [   38     0]]

 [[11274    98]
  [  289   307]]

 [[11869     3]
  [   79    17]]]
```

5.10 Model Saving

```
# selecting the best model
best_model = trained_models['Support Vector Classifier']['trained']

# saving the best classification model
import joblib
joblib.dump(best_model,open('Malignant_comments_classifier.pkl','wb'))
```

5.11 Predictions on test data

```
# saving the best classification model
import joblib
best_model = joblib.load('Malignant_comments_classifier.pkl')
#joblib.dump(best_model,open('Malignant_comments_classifier.pkl','wb'))
```

```
test_vec.toarray()
```

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

```
# Make predictions and view the results
predict_test = best_model.predict(test_vec.toarray())

# Saving predicted values into a CSV file
pd.DataFrame(predict_test).to_csv('Predicted_test_output.csv')
```

```
df1 = pd.read_csv('Predicted_test_output.csv')
df1.drop("Unnamed: 0", axis=1, inplace=True)
df1.rename({'0':'malignant', '1':'highly_malignant', '2':'rude', '3':'threat', '4':'abuse', '5':'loathe'},
           axis='columns', inplace=True)
df2=df_test.copy()
df = pd.concat([df2, df1], axis=1)
df
```

	comment_text	length_before_cleaning	length_after_cleaning	malignant
0	yo bitch ja rule succesful ever whats hating s...	367	235	NaN
1	rfc title fine imo	50	18	NaN
2	source zawe ashton lapland	54	26	NaN
3	look back source information updated correct f...	205	109	NaN
4	anonymously edit article	41	24	NaN
...	...	...	...	...
153159	totally agree stuff nothing long crap	60	37	NaN
153160	throw field home plate get faster throwing cut...	198	107	NaN
153161	okinotorishima category see change agree corre...	423	238	NaN
153162	one founding nation eu germany law return quit...	502	319	NaN
153163	stop already bullshit welcome fool think kind ...	141	74	NaN

153164 rows × 4 columns

5.12 Saving predictions in a csv file

```
1 df.to_csv('test_dataset_predictions2.csv', index=False)
```

Chapter 6

Conclusion

- Linear Support Vector Classifier performs better with Accuracy Score: 91.15077857956704 % and Hamming Loss: 2.0952019242942144 % than the other classification models.
- Final Model (Hyperparameter Tuning) is giving us Accuracy score of 91.26% which is slightly improved compare to earlier Accuracy score of 91.15%.
- Final Model (Hyperparameter Tuning) is giving us Accuracy score of 91.26% which is slightly improved compare to earlier Accuracy score of 91.15%.

Limitations of this work and Future Scope

- 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.