MALIGNANT COMMENTS CLASSIFICATION PROJECT

Submitted By :

Milind B. Kuwar

ACKNOWLEDGMENT

The project would not have been built without the constant support from DataTrained and Fliprobo teams.

References:

1. Notes and classes by DataTrained Academy

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.

Sentiment Analysis is the task of analyzing people's opinions in textual data (e.g., product reviews, movie reviews, or tweets), and extracting their polarity and viewpoint. The task can be cast as either a binary or a multi-class problem. Binary sentiment analysis classifies texts into positive and negative classes, while multi-class sentiment analysis classifies texts into fine-grained labels or multi-level intensities.

The goal of this project is to use deep learning techniques to identify the level toxicity of a comment which could be used to help deter users from posting potentially hurtful comments, engage in more sophisticated arguments and make internet a safer place.

Our dataset consists of 6 labels namely highly malignant, malignant, abuse, threat, loathe and rude. This project aims to implement various Machine Learning algorithms and deep learning algorithms like Multilayer perceptron(MLP), Long Short Term Memory Networks, Multinomial Naïve Baiyes, Logistic Regression, Random Forest Classifier , Linear SVC and Adaptive Boosting.

- ## Conceptual Background of the Domain Problem

  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 in-offensive, but “u are an idiot” is clearly offensive.
  The above problem statement clearly explains that the target variable is not continuous & it’s a classification problem as we need to predict the level toxicity of a comment. So this can be solved by Classification Machine learning algorithms:

Analytical Problem Framing

• Analytical Modelling of the Problem

Methodology represents a description about the framework that is undertaken. It consists of various milestones that need to be achieved in order to fulfil the objective. We have various attributes that contribute retaining customers that aids the growth of business.

With continuous increase in available data, there is a pressing need to organize it and modern classification problems often involve the prediction of multiple labels simultaneously associated with a single instance. Known as Multi-Label Classification, it is one such task which is omnipresent in many real world problems. In this project also, we have multi-label classification problem.

We have used Tf-Idf Vectorizer to vectorize the words in our dataset. TF-IDF is an abbreviation for Term Frequency Inverse Document Frequency. This is very common algorithm to transform text into a meaningful representation of numbers which is used to fit machine algorithm for prediction. It is very important for tuning performance on NLP projects.

The following steps represents stepwise tasks that need to be completed:

- Data Collection
- Exploratory Data Analysis
- Data pre-processing
- Data Visualization
- Model Building
- Model Evaluation
- Saving the best model

• Data Sources and their formats

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. The sample data for the reference is as shown below:
  Data Types : int64, float64, object

```
1 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'awww! 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

```
1 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.

Data Preprocessing

- **Data Preprocessing Done**

  - Checked for null values and identified no null values in dataset.
  - Outliers and skewness were checked.

```
# Checking the shape of the train dataset
df_train.shape
```

```
(159571, 8)
```

```
# Checking the shape of the test dataset
df_test.shape
```

```
(153164, 2)
```

```
# Checking the number of unique comments
df_train.nunique()
```

```
id                  159571
comment_text        159571
malignant                2
highly_malignant         2
rude                     2
threat                   2
abuse                    2
loathe                   2
dtype: int64
```

```
# Checking the missing values in the dataset
df_test.isnull().sum()
```

```
id              0
comment_text    0
dtype: int64
```

```
# Checking the missing values in the dataset
df_train.isnull().sum()
```

```
id                  0
comment_text        0
malignant           0
highly_malignant    0
rude                0
threat              0
abuse               0
loathe              0
dtype: int64
```

For cleaning the data firstly the data was checked for inconsistencies by checking the standard deviation. On finding higher std. values it could be guessed that there were outliers present in the data. In order to get a better understanding of the data, we plotted distribution of data. We noticed that the dataset had some outliers, which was negligible.

```
# Checking the inormation of the dataset
df_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 159571 entries, 0 to 159570
Data columns (total 8 columns):
 #   Column            Non-Null Count   Dtype 
---  ------            --------------   ----- 
 0   id                159571 non-null  object
 1   comment_text      159571 non-null  object
 2   malignant         159571 non-null  int64 
 3   highly_malignant  159571 non-null  int64 
 4   rude              159571 non-null  int64 
 5   threat            159571 non-null  int64 
 6   abuse             159571 non-null  int64 
 7   loathe            159571 non-null  int64 
dtypes: int64(6), object(2)
memory usage: 9.7+ MB
```

Observation:

1-Information shows that there are no null values present in our dataset.
2-ID and Comment_text feature is only object datatype and rest all the features are integer datatype.

- Converting all message to lower case
- Replace email address with 'email'
- Replace money symbol with 'moneysymb'(£ can type with ALT key+156)
- Replace 10 digit phone number(format include paranthesis, space, no spaces,dashes) with 'phone number'
- Remove punctuation
- Remove leading and trailing whitespace
- Remove leading and trailing whitespace
- Replace whitespace between terms with a single space
- Remove stopwords
-

```
#converting all message to lower case
df_train['comment_text']=df_train['comment_text'].str.lower()
```

```
#Replace email address with 'email'
df_train['comment_text']=df_train['comment_text'].str.replace(r'^.+@[^\.].*\[a-z]{2,}$','emailaddress')
#Replace URLs with 'webaddress'
df_train['comment_text']=df_train['comment_text'].str.replace(r'^http\://[a-zA-Z0-9\-\.]+\.[a-zA-Z]{2,3}(/\s*)?$','webaddres
#Replace money symbol with 'moneysymb'(£ can type with ALT key+156)
df_train['comment_text']=df_train['comment_text'].str.replace(r'£|\$','dollers')
#Replace 10 digit phone number(format include paranthesis, space, no spaces,dashes) with 'phone number'
df_train['comment_text']=df_train['comment_text'].str.replace(r'^\(?[\d]{3}\)?[\s-]?[\d]{3}[\s-]?[\d]{4}$','phonenumber')
#Replace number with 'numbr'
df_train['comment_text']=df_train['comment_text'].str.replace(r'^\d+(\.\d+)?','numbr')
```

```
#Remove punctuation
df_train['comment_text']=df_train['comment_text'].str.replace(r'[^\w\d\s]',' ')
#replace whitespace between terms with a single space
df_train['comment_text']=df_train['comment_text'].str.replace(r'\s+',' ')
#remove leading and trailing whitespace
df_train['comment_text']=df_train['comment_text'].str.replace(r'^\s+|\s+?$',' ')
```

```
#remove stopwords
import string
import nltk
from nltk.corpus import stopwords
stop_words = set(stopwords.words('english') + ['u', 'ü', 'ur', '4', '2', 'im', 'dont', 'doin', 'ure'])

df_train['comment_text']=df_train['comment_text'].apply(lambda x:' '.join(term for term in x.split() if term not in stop_wor
```

After performing all the above steps and also adding a new feature to check new comment length after cleaning, our dataset would look as follows:

```
df_train['clean_length']=df_train.comment_text.str.len()
df_train.head()
```

	comment_text	malignant	highly_malignant	rude	threat	abuse	loathe	length	clean_length
0	explanation edits made username hardcore metal...	0	0	0	0	0	0	264	171
1	aww matches background colour seemingly stuck ...	0	0	0	0	0	0	112	83
2	hey man really trying edit war guy constantly ...	0	0	0	0	0	0	233	141
3	make real suggestions improvement wondered sec...	0	0	0	0	0	0	622	374
4	sir hero chance remember page	0	0	0	0	0	0	67	29

- # Hardware and Software Requirements and Tools Used

Listing down the hardware and software requirements along with the tools, libraries and packages used. Describe all the software tools used along with a detailed description of tasks done with those tools.

2. Lenovo ideapad145 laptop
3. Jupyter Notebook - The Jupyter Notebook is an interactive environment for running code in the browser. It is a great tool for exploratory data analysis and is widely used by data scientists.
4. MS powerpoint - For preparing presentation of project.
5. MS word - For preparing report
6. Matplotlib - For data visualization to produce high quality plots, charts and graphs.
7. Pandas - Python library for data wrangling and analysis. It is built around a data structure called DataFrame.
8. Numpy - NumPy is one of the fundamental packages for scientific computing in Python.

Model/s Development and Evaluation

- ## Identification of possible problem-solving approaches (methods)

  I have use statistical models to find insights given set of data. Predictions are assessed differently depending on its type: Based on the data structure of Target variable, Model type was identified

  First and important step was to identify the model which was done by analysing the target variable . Best Acuuracy score was achieved using Logistic Regression

- ## Testing of Identified Approaches (Algorithms)

  Build Models based on learning it was a supervised classification problem.
  I built 5 models to evaluate performance of each of them:

  1. **Logistic Regression**
  2. **MultinomialNB**
  3. **KNeighbors Classifier**
  4. **DecisionTree Classifier**
  5. **SVC**

- **Run and Evaluate selected models**

  Describing all the algorithms used along with the snapshot of their code and what were the results observed over different evaluation metrics.

Logistic Regression

```
************** LogisticRegression ***************


LogisticRegression()


Accuracy_score= 0.9531667780748663


Cross_Val_Score= 0.9535128562209592


roc_auc_score= 0.7924918146002661


classification report
              precision    recall  f1-score   support

           0       0.96      0.99      0.97     43004
           1       0.92      0.59      0.72      4868

    accuracy                           0.95     47872
   macro avg       0.94      0.79      0.85     47872
weighted avg       0.95      0.95      0.95     47872



[[42754   250]
 [ 1992  2876]]


AxesSubplot(0.125,0.808774;0.62x0.0712264)
```

MultinomialNB

```
************** MultinomialNB ***************


MultinomialNB()


Accuracy_score= 0.9354737633689839


Cross_Val_Score= 0.9367303554748633


roc_auc_score= 0.6884622511658735


classification report
               precision    recall  f1-score   support

           0       0.93      1.00      0.97     43004
           1       0.97      0.38      0.54      4868

    accuracy                           0.94     47872
   macro avg       0.95      0.69      0.75     47872
weighted avg       0.94      0.94      0.92     47872


[[42941    63]
 [ 3026  1842]]


AxesSubplot(0.125,0.808774;0.62x0.0712264)
```

Decision Tree Classifier

```
************** DecisionTreeClassifier ***************


DecisionTreeClassifier()


Accuracy_score= 0.9392337901069518


Cross_Val_Score= 0.9399076251956352


roc_auc_score= 0.8263624575788062


classification report
               precision    recall  f1-score   support

           0       0.96      0.97      0.97     43004
           1       0.71      0.68      0.70      4868

    accuracy                           0.94     47872
   macro avg       0.84      0.83      0.83     47872
weighted avg       0.94      0.94      0.94     47872


[[41630  1374]
 [ 1535  3333]]


AxesSubplot(0.125,0.808774;0.62x0.0712264)
```

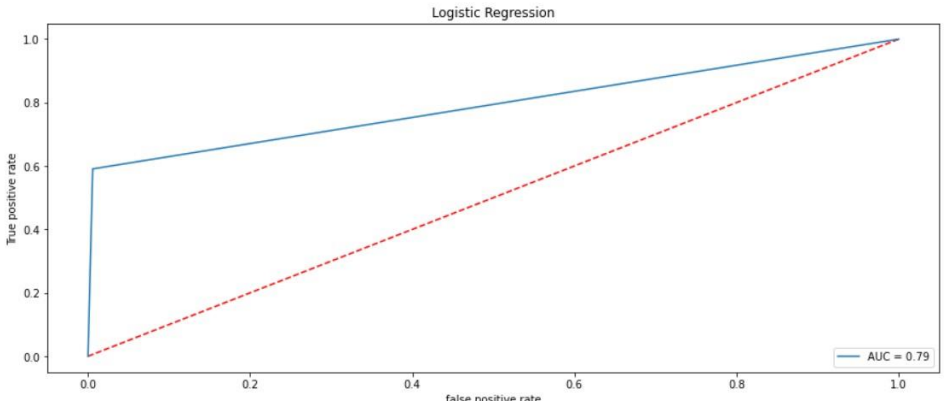
KNeighbors Classifier

```
************** KNeighborsClassifier ***************

KNeighborsClassifier()

Accuracy_score= 0.8963277072192514

Cross_Val_Score= 0.8967356214680471

roc_auc_score= 0.6214955391587276

classification report
               precision    recall  f1-score   support

           0       0.92      0.97      0.94     43004
           1       0.48      0.28      0.35      4868

    accuracy                           0.90     47872
   macro avg       0.70      0.62      0.65     47872
weighted avg       0.88      0.90      0.88     47872


[[41563  1441]
 [ 3522  1346]]

AxesSubplot(0.125,0.808774;0.62x0.0712264)
```

Since the dataset was too large it took me around 9-10 hours to get these results for these algorithms. Hence I had to interrupt the kernel and proceed with available results. Out of all the algorithms, Logistic Regression was giving best score. Also, its cross validation score was also satisfactory. Its ROC_AUC curve is as shown:

```
# Roc-Auc score
f,ax = plt.subplots(figsize = (15,6))
# Calculate fpr, tpr and thresholds
fpr, tpr, thresholds = roc_curve(y_test, pred)
ax.plot([0,1],[0,1],'r--')
ax.plot(fpr,tpr,label='AUC = %0.2f'% roc_auc_score(y_test, pred))
ax.legend(loc='lower right')
ax.set_xlabel('false positive rate')
ax.set_ylabel('True positive rate')
ax.set_title('Logistic Regression')
```

```
Text(0.5, 1.0, 'Logistic Regression')
```



Predicting Test Dataset

```
def Tf_idf_test(text):
    tfid = TfidfVectorizer(max_features=43194,smooth_idf=False)
    return tfid.fit_transform(text)
```

```
x_test_data=Tf_idf_test(test['clean_comment_text'])
```

```
x_test_data.shape
```

```
(153164, 43194)
```

```
Prediction=LR.predict(x_test_data)
test['Predicted Labels']=Prediction
test
```

	id	comment_text	comment_length	clean_comment_text	clean_comment_length	Predicted Labels
0	00001cee341fdb12	Yo bitch Ja Rule is more succesful then you'll...	367	bitch rule succesful whats hating mofuckas bit...	184	0
1	0000247867823ef7	== From RfC == \n\n The title is fine as it is...	50	title fine	10	0
2	00013b17ad220c46	" \n\n == Sources == \n\n * Zawe Ashton on Lap...	54	source zawe ashton lapland	26	0
3	00017563c3f7919a	:If you have a look back at the source, the in...	205	look source information updated correct form g...	109	0
4	00017695ad8997eb	I don't anonymously edit articles at all.	41	anonymously edit article	24	0
...	...	...	...	...	...	...
153159	fffcd0960ee309b5	. \n i totally agree, this stuff is nothing bu...	60	totally agree stuff long crap	29	0
153160	fffd7a9a6eb32c16	== Throw from out field to home plate. == \n\n...	198	throw field home plate faster throwing direct ...	85	0
153161	fffda9e8d6fafa9e	" \n\n == Okinotorishima categories == \n\n I ...	423	okinotorishima category change agree correct g...	212	0
153162	fffe8f1340a79fc2	" \n\n == ""One of the founding nations of the...	502	founding nation germany return similar israel ...	275	0
153163	ffffce3fb183ee80	" \n :::Stop already. Your bullshit is not wel...	141	stop bullshit welcome fool think kind explinat...	54	0

153164 rows × 6 columns

Saving Model:

```
1 test['Predicted Labels'].value_counts()
```

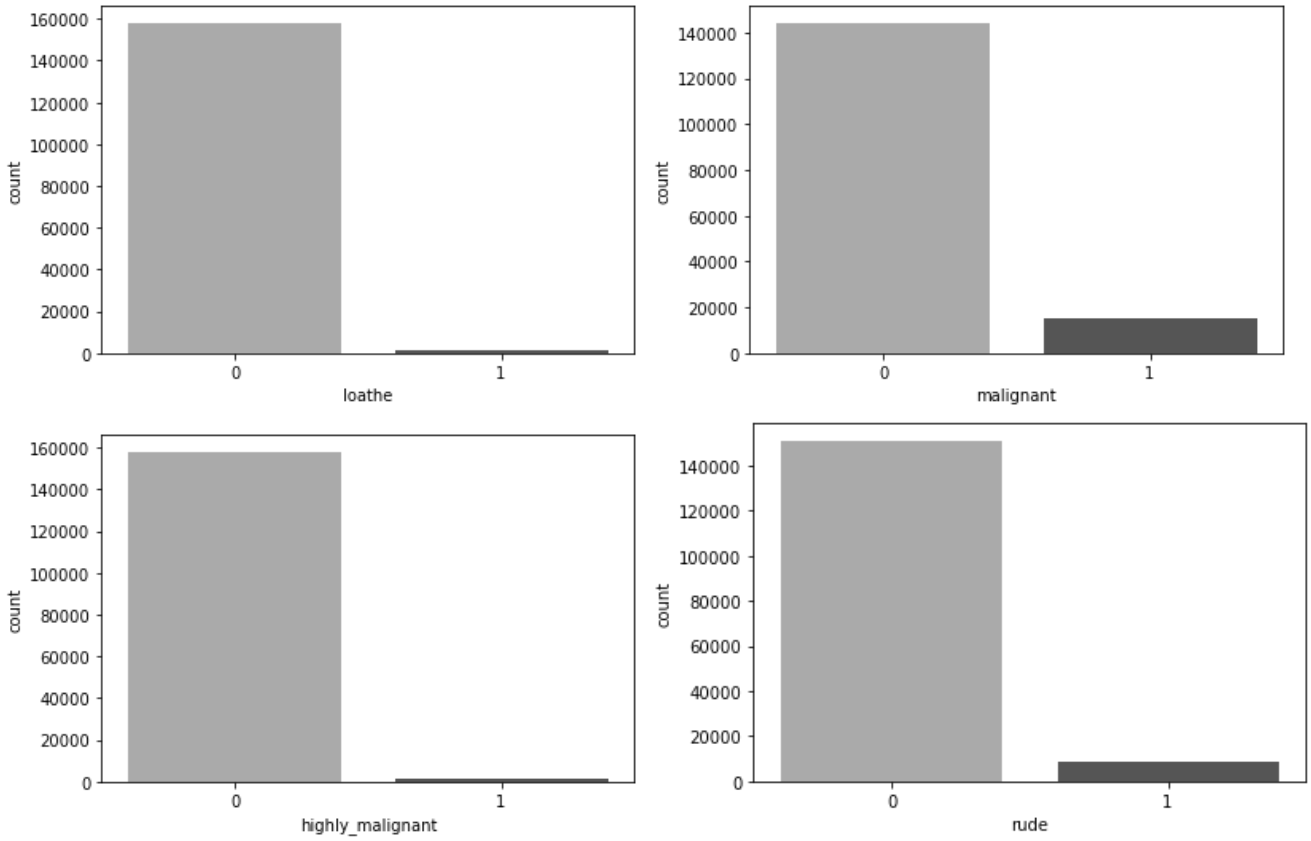
```
0    152452
1       712
Name: Predicted Labels, dtype: int64
```

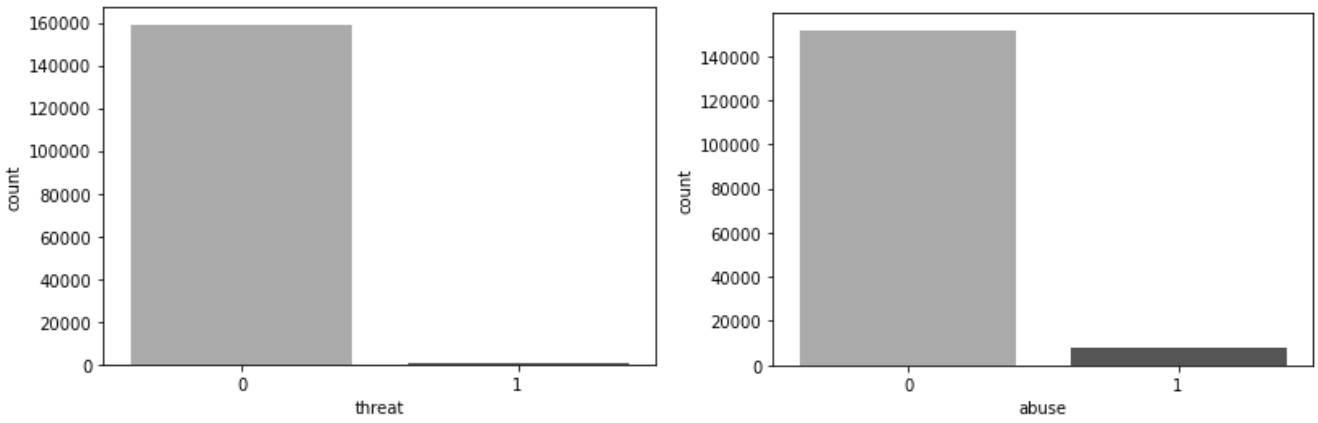
```
1 # Pickle file.
2 import joblib
3 joblib.dump(LR,'Malignant_Prediction.pkl')
```

```
['Malignant_Prediction.pkl']
```

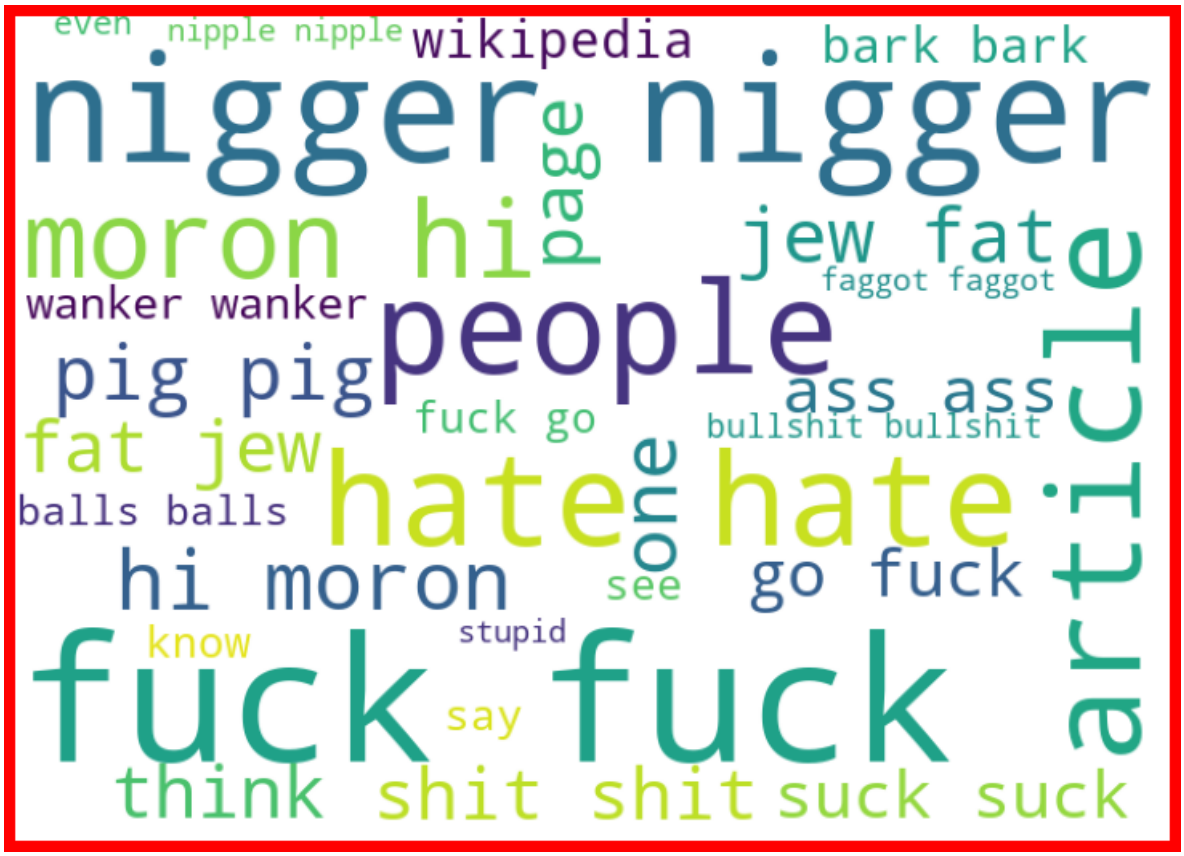
```
1 test.to_csv('Malignant_Predict.csv')
```

- **Visualizations & Interpretation of the Results**

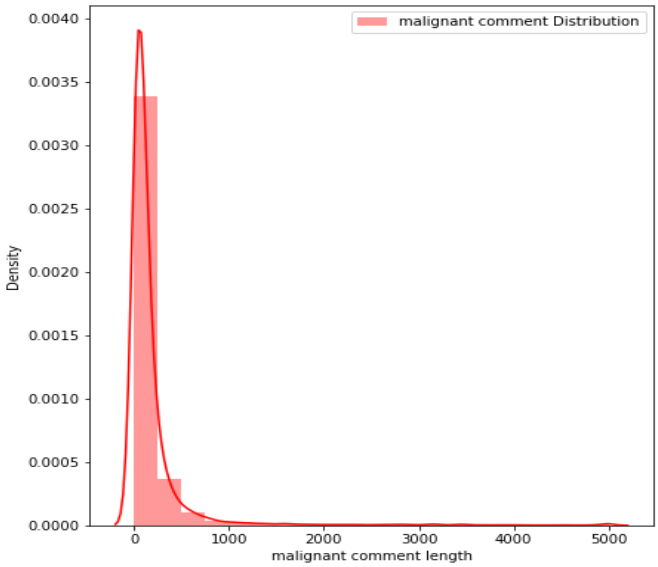


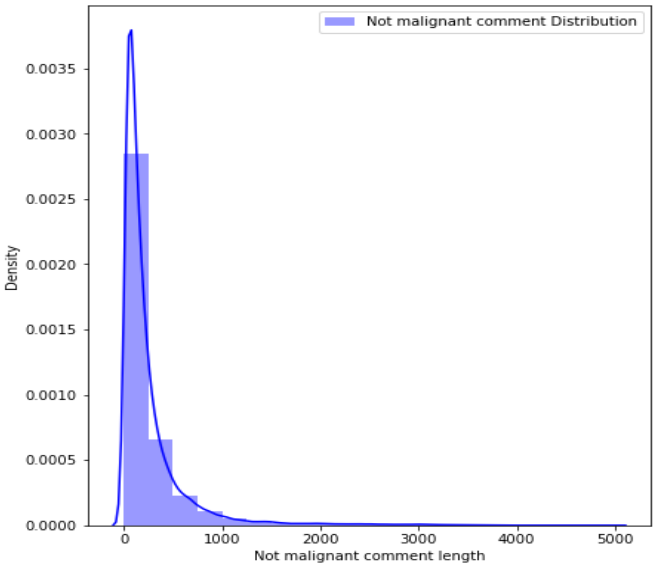


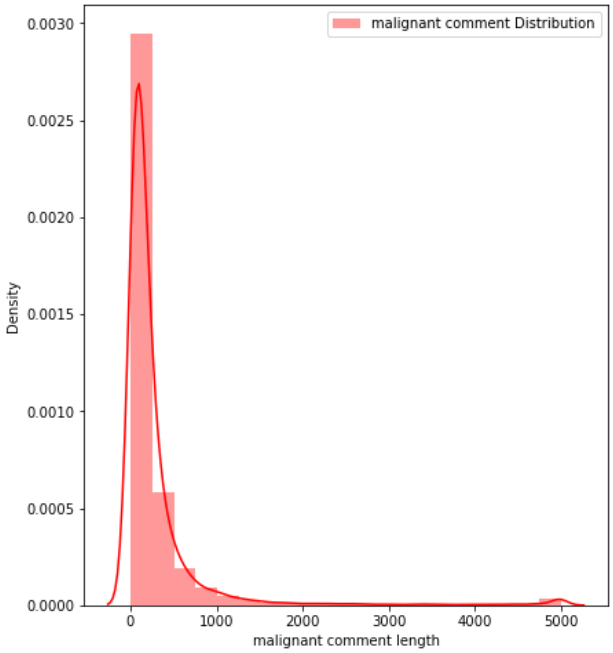
We have also observed most frequent words in positive and negative comments through word-cloud:

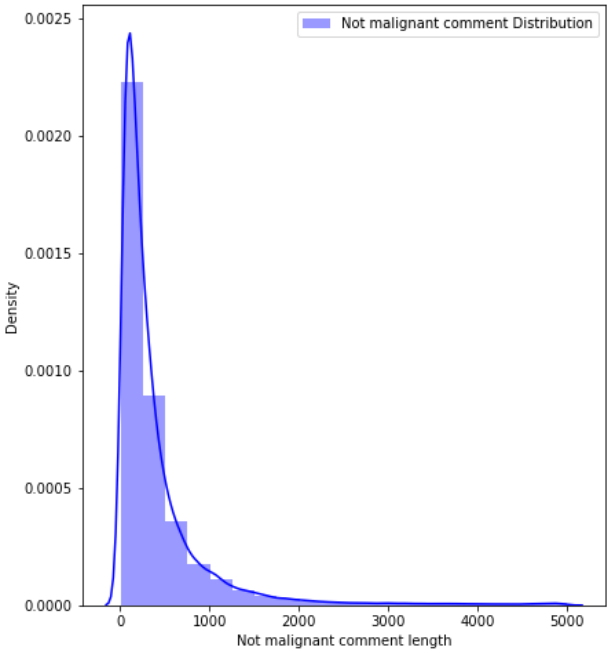


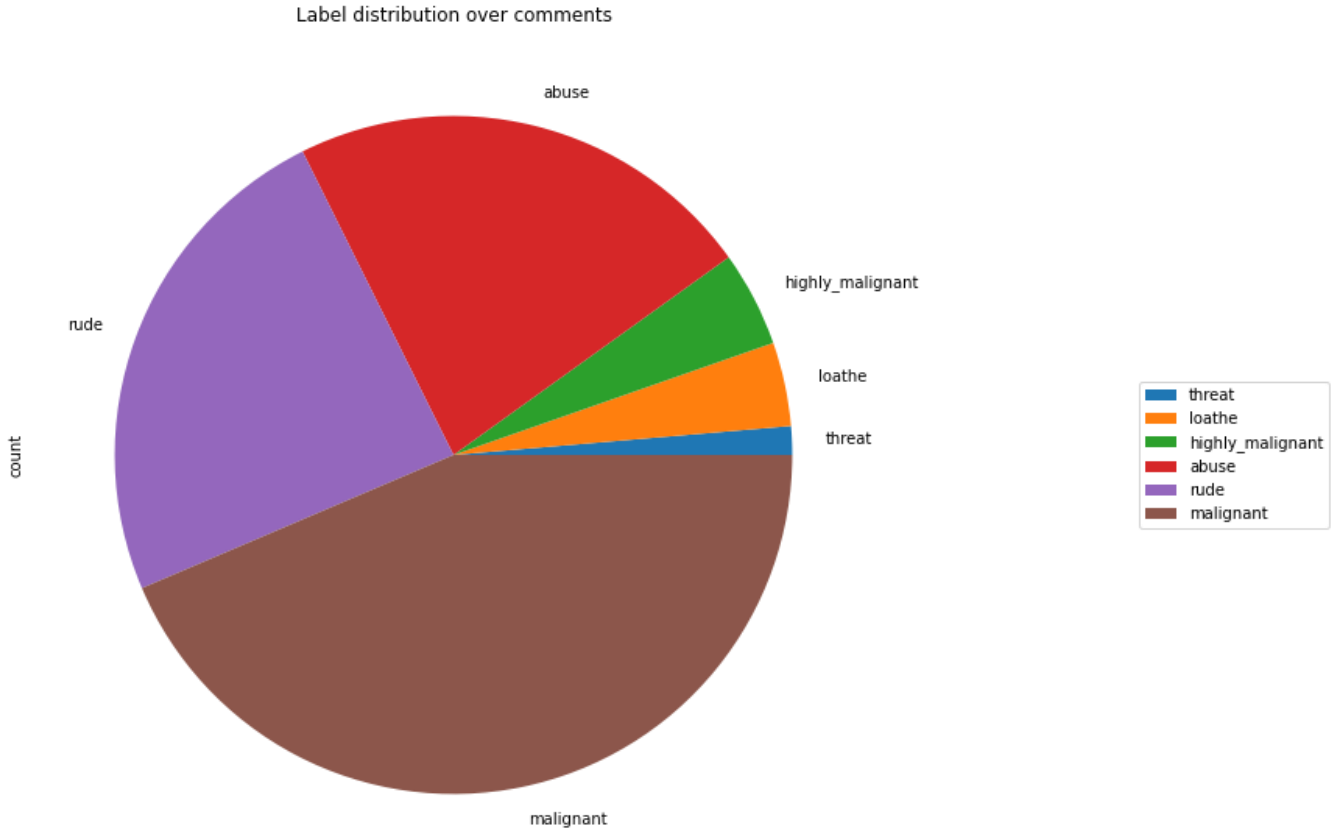
Then we have checked distribution of comment length before and after cleaning.






malignant comment Distribution
Density
malignant comment length
0.0030
0.0025
0.0020
0.0015
0.0010
0.0005
0.0000
0
1000
2000
3000
4000
5000


Not malignant comment Distribution
Density
Not malignant comment length
0.0025
0.0020
0.0015
0.0010
0.0005
0.0000
0
1000
2000
3000
4000
5000


Label distribution over comments
abuse
highly_malignant
rude
loathe
threat
count
malignant
threat
loathe
highly_malignant
abuse
rude
malignant

CONCLUSION

- We have got Logistic Regression as best model since it's giving us good result and other metrics are also satisfactory.
- Using Logistic Regression as our final algorithm we have predicted the values for test dataset and it's also working well and is able to differentiate/predict negative comments and non-negative (good) comments.
- From displaying the data, it seems there is lot of special characters present in the data. So, it is better to proceed by filter it out.
- As the above data is in text, so presence of special characters and stopwords is always there.
- After proper cleaning and processing, decision tree classifier gives the highest accuracy as well as ROC Score.

Limitations of this work and Scope for Future Work

- Multinomial naïve bayes and Random Forest using hyperparameter tuning.
- After analyzing for each behaviour separately by creating models.
- Support Vector Machine also performs well in text data but its hyper-parameter tuning is very complex and takes much more time