



Malignant Comments Classification



Submitted by:

SANTOSH H. HULBUTTI

Table of Contents

ACKNOWLEDGMENT.....	3
INTRODUCTION	4
<i>Business Problem Framing</i>	4
<i>Conceptual Background of the Domain Problem</i>	4
<i>Review of Literature</i>	4
<i>Motivation for the Problem Undertaken</i>	4
ANALYTICAL PROBLEM FRAMING	5
<i>Mathematical/ Analytical Modeling of the Problem</i>	5
<i>Data Sources and their formats</i>	5
<i>Data Pre-processing Done</i>	6
<i>Data Inputs- Logic- Output Relationships</i>	7
<i>State the set of assumptions (if any) related to the problem under Consideration</i>	7
<i>Hardware and Software Requirements and Tools Used</i>	7
MODEL DEVELOPMENT AND EVALUATION	8
<i>Identification of possible problem-solving approaches (methods).....</i>	8
<i>Testing of Identified Approaches (Algorithms).....</i>	8
<i>Run and evaluate selected models.....</i>	9
<i>Key Metrics for success in solving problem under consideration.....</i>	14
<i>Hyperparameter Tuning:</i>	14
<i>Saving & predictions of the model on Test data provided</i>	15
VISUALIZATIONS & EDA	17
CONCLUSION	23
<i>Key Findings and Conclusions of the Study</i>	23
<i>Learning Outcomes of the Study in respect of Data Science</i>	23
<i>Limitations of this work and Scope for Future Work</i>	23

ACKNOWLEDGMENT

This project is completed using knowledge/information available on internet.

Following are the websites & YouTube Channels, which were used to scrape data & understand concepts related to ML, AI & Data Visualization.

Websites:

1. towardsdatascience.com
2. medium.com
3. analyticsvidya.com
4. DataTrained LMS Platform
5. Carwale.com
6. Official documentation of ScikitLearn, Matplotlib library, AutoViz, Sweet Viz, Pandas Library & Seaborn library.
7. Kaggle.com
8. UCI ML Repository
9. Stackoverflow.com
10. YouTube Channels:
 - a. Krish Naik
 - b. Nicholas Renotte
 - c. Sidhdhardan
 - d. Keith Galli

I would like to thank FlipRobo Technologies, for giving an opportunity to work as an intern during this project period. And also like to thank mentor Ms. Gulshana Chaudhary for assigning the project.

INTRODUCTION

Business Problem Framing

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

Online hate, described as abusive language, aggression, 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.

Conceptual Background of the Domain Problem

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.

So, we build a machine learning model that helps to understand which comment is malignant and which one is being based on the sample data. We do Exploratory Data Analysis to visualize the data graphically which is easy to understand. We also used some statistical techniques to see the insights of the data.

Review of Literature

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 be used to classify hate and offensive comments so that it can be controlled and restricted from spreading hatred and cyberbullying. First, we do all the data pre-processing steps and do EDA to visualize the data graphically and after that we make a machine learning model in order to improve.

Motivation for the Problem Undertaken

Every investigation begins with ideas that are further developed and inspired to address a variety of situations and circumstances. The client wants some predictions that could help them in further investment and improvement in selection of comments. So, to help them we make this project. My motivation behind this project is to do the proper research because research is a process for finding a solution to a problem after making a deep analysis and conducting studies of relevant factors. In general, research is a method designed to ensure that the information obtained is reasonable and supported by the quantitative and qualitative data, and that involves a systematic process. It includes the process of designing research methods, collecting and describing.

ANALYTICAL PROBLEM FRAMING

Mathematical/ Analytical Modeling of the Problem

- First, we check the basic information of the dataset that is its shape and its information using pandas library. The basic information tells the data type of our column and number of data present.
- Then we check the unique information of our data columns.
- After that we check for null values, if it present then we remove it because our data is text data and we cannot fill it.
- After that we check the summary statistics of our dataset. This part talks about the statistics of our dataset i.e., mean, median, max value, min values and also it tells whether outliers are present in our dataset or not.
- We check the length of data to understand the distribution of data.
- We use seaborn library to plot the target data and using word cloud for getting the sense of loud words in Malignant and Benign comments.
- Similarly, we can also make word cloud for separate columns where you can check the loud words for particular features because there are six target features.
- Before making the model, we convert our text into vectors so for that we use technique known as TF-IDF Vectorizer

Data Sources and their formats

- In this project the sample data is provided to us from our client database. The dataset is in csv (comma separated values) format.
- 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.

Data Pre-processing Done

1. First, we check the information of the given dataset because it tells that how many rows and columns are present in our dataset and data type of the columns whether they are object, integer or float.
2. Dropping duplicates rows if present in dataset.
3. Then we check for the null values present in our dataset. If null values are present then drop it because we cannot able to fill the text data.
4. After that we check the summary statistics of our dataset. This part talks about the statistics of our dataset i.e., mean, median, max value, in values and also it tells whether outliers are present in our dataset or not.
5. We also check the correlation of our target features with each other. If columns are highly correlated with each other let's say 90% or above then remove those columns to avoid multicollinearity problem.
6. There are six features in this dataset so we combined them and make one target feature giving the name as label and also, we do scale for that feature.
7. In data cleaning we use mainly five steps using function:
 - a. Removing HTML tags
 - b. Removing special characters
 - c. Converting everything to lowercase
 - d. Removing stop words
 - e. Using WordNet Lemmatization for lemmatization
8. We create new column (clean text) after removing punctuations, stop words from input feature to check how much data is cleaned.

Data Inputs- Logic- Output Relationships

If the input has a particular word or relational words then, the value of the label's changes into 1, otherwise 0.

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

- All stop words are present inside the stop words library
- The offensive words are always explicit and sarcasms are not considered

Hardware and Software Requirements and Tools Used

a. Software

- Jupyter Notebook (Python 3.9)
- Microsoft Office

b. Hardware

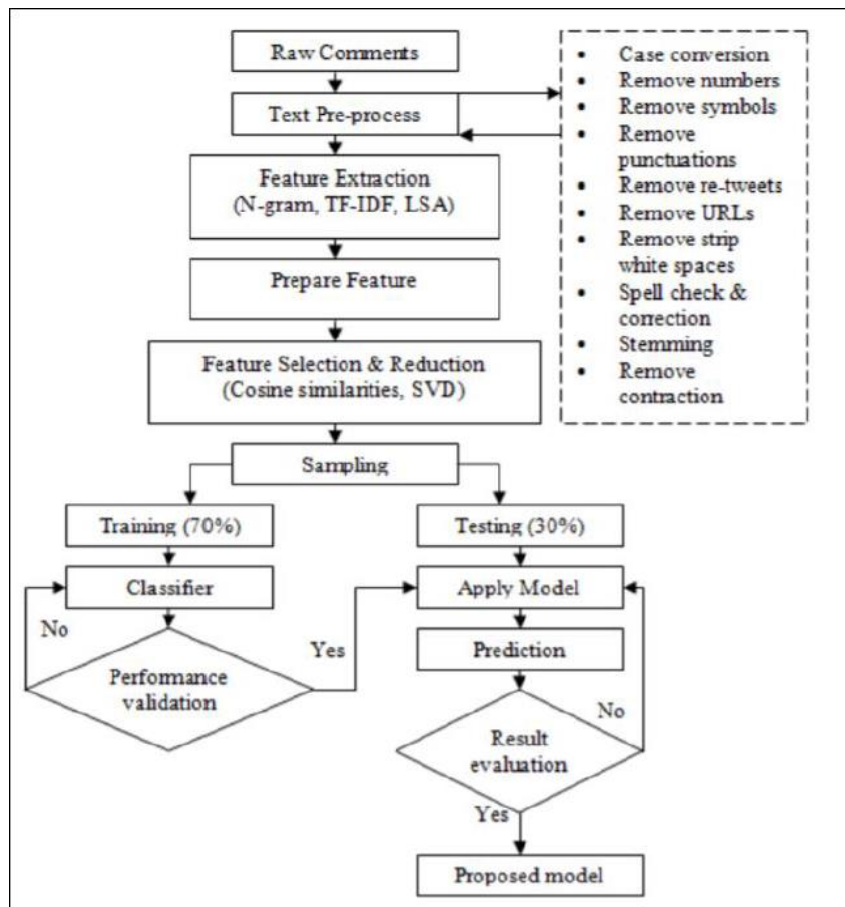
- Processor – AMD Ryzen 5
- RAM - 8 GB
- Graphic Memory - 4Gb, Nvidia GEFORCE RTX1650

c. Python Libraries

- Pandas
- Numpy
- Selenium
- Matplotlib
- Seaborn
- Scipy
- Sklearn & skmultilearn
- NLTK
- re
- Gradio

MODEL DEVELOPMENT AND EVALUATION

Identification of possible problem-solving approaches (methods)



Testing of Identified Approaches (Algorithms)

```
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import MultinomialNB
from xgboost import XGBClassifier
from sklearn.svm import LinearSVC
from lightgbm import LGBMClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import SGDClassifier
from sklearn.multiclass import OneVsRestClassifier
```

	models_name	accuracies_train	accuracies_test	precision_scores	f1_scores	hamming_losses
5	KNeighbors Classifier	91.27	89.19	67.42	48.65	2.94
1	Multinomial Naive Bayes	91.55	91.26	87.56	60.47	2.21
6	Random Forest Classifier	99.10	91.29	80.18	67.88	2.04
7	StochasticGB Classifier	91.67	91.47	88.73	62.85	2.10
4	LightGBM Classifier	93.10	91.60	81.86	68.20	1.99
2	XGB Classifier	93.54	91.65	83.84	67.12	2.01
3	Linear SVC	92.28	91.74	84.71	67.89	1.96
0	Logistic Regression	92.12	91.76	86.02	66.65	1.99

Run and evaluate selected models

```
lgr = LogisticRegression(solver='lbfgs')
mnb = MultinomialNB()
xgb = XGBClassifier()
lsvc = LinearSVC(max_iter = 3000)
lgbm = LGBMClassifier()
knc = KNeighborsClassifier()
rfc = RandomForestClassifier()
sgdc = SGDClassifier()
```

```
models = [lgr,mnb,xgb,lsvc,lgbm,knc,rfc,sgdc]
models_name = ['Logistic Regression','Multinomial Naive Bayes',
               'XGB Classifier','Linear SVC','LightGBM Classifier',
               'KNeighbors Classifier','Random Forest Classifier','StochasticGB Classifier']
x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size = 0.25, random_state = 42)
```

for Logistic Regression model..

Accuracy score for Train : 92.12%
Accuracy score for Test : 91.76%
F1 score for Test : 66.65%
Precision for Test : 86.02%
Hamming Loss : 1.99%

Confusion Matrix:

```
[[[35791  247]
   [ 1609 2218]]
```

```
[[[39424  52]
   [ 306  83]]
```

```
[[[37593  149]
   [ 776 1347]]
```

```
[[[39723  15]
   [ 113  14]]
```

```
[[[37663  270]
   [ 922 1010]]
```

```
[[[39470  41]
   [ 265  89]]]
```

Test Classification report:

	precision	recall	f1-score	support
0	0.90	0.58	0.71	3827
1	0.61	0.21	0.32	389
2	0.90	0.63	0.74	2123
3	0.48	0.11	0.18	127
4	0.79	0.52	0.63	1932
5	0.68	0.25	0.37	354
micro avg	0.86	0.54	0.67	8752
macro avg	0.73	0.39	0.49	8752
weighted avg	0.85	0.54	0.66	8752
samples avg	0.05	0.05	0.05	8752

for Multinomial Naive Bayes model..

Accuracy score for Train : 91.55%
Accuracy score for Test : 91.26%
F1 score for Test : 60.47%
Precision for Test : 87.56%
Hamming Loss : 2.21%

Confusion Matrix:

```
[[[35914  124]
 [ 1976 1851]]

 [[39401   75]
 [  287  102]]

 [[37618  124]
 [  977 1146]]

 [[39738    0]
 [  127    0]]

 [[37702  231]
 [ 1037  895]]

 [[39491   20]
 [  306   48]]]
```

Test Classification report:

	precision	recall	f1-score	support
0	0.94	0.48	0.64	3827
1	0.58	0.26	0.36	389
2	0.90	0.54	0.68	2123
3	0.00	0.00	0.00	127
4	0.79	0.46	0.59	1932
5	0.71	0.14	0.23	354
micro avg	0.88	0.46	0.60	8752
macro avg	0.65	0.31	0.41	8752
weighted avg	0.86	0.46	0.60	8752
samples avg	0.04	0.04	0.04	8752

for XGB Classifier model..

Accuracy score for Train : 93.54%
Accuracy score for Test : 91.65%
F1 score for Test : 67.12%
Precision for Test : 83.84%
Hamming Loss : 2.01%

Confusion Matrix:

```
[[[35748  290]
 [ 1636 2191]]

 [[39391   85]
 [  308   81]]

 [[37557  185]
 [  701 1422]]

 [[39718   20]
 [  106   21]]

 [[37612  321]
 [  852 1080]]

 [[39468   43]
 [  251  103]]]
```

Test Classification report:

	precision	recall	f1-score	support
0	0.88	0.57	0.69	3827
1	0.49	0.21	0.29	389
2	0.88	0.67	0.76	2123
3	0.51	0.17	0.25	127
4	0.77	0.56	0.65	1932
5	0.71	0.29	0.41	354
micro avg	0.84	0.56	0.67	8752
macro avg	0.71	0.41	0.51	8752
weighted avg	0.83	0.56	0.67	8752
samples avg	0.05	0.05	0.05	8752

for Linear SVC model..

Accuracy score for Train : 92.28%
Accuracy score for Test : 91.74%
F1 score for Test : 67.89%
Precision for Test : 84.71%
Hamming Loss : 1.96%

Confusion Matrix:

```
[[[35728  310]
 [ 1519 2308]]

 [[39439   37]
 [  320   69]]

 [[37571  171]
 [  723 1400]]

 [[39722   16]
 [  100   27]]

 [[37626  307]
 [  872 1060]]

 [[39457   54]
 [  261   93]]]
```

Test Classification report:

	precision	recall	f1-score	support
0	0.88	0.60	0.72	3827
1	0.65	0.18	0.28	389
2	0.89	0.66	0.76	2123
3	0.63	0.21	0.32	127
4	0.78	0.55	0.64	1932
5	0.63	0.26	0.37	354
micro avg	0.85	0.57	0.68	8752
macro avg	0.74	0.41	0.51	8752
weighted avg	0.84	0.57	0.67	8752
samples avg	0.05	0.05	0.05	8752

for LightGBM Classifier model..

Accuracy score for Train : 93.10%
Accuracy score for Test : 91.60%
F1 score for Test : 68.20%
Precision for Test : 81.86%
Hamming Loss : 1.99%

Confusion Matrix:

```
[[[35699  339]
 [ 1573 2254]]

 [[39398   78]
 [  301   88]]

 [[37529  213]
 [  653 1470]]

 [[39699   39]
 [   98   29]]

 [[37516  417]
 [  757 1175]]

 [[39463   48]
 [  254  100]]]
```

Test Classification report:

	precision	recall	f1-score	support
0	0.87	0.59	0.70	3827
1	0.53	0.23	0.32	389
2	0.87	0.69	0.77	2123
3	0.43	0.23	0.30	127
4	0.74	0.61	0.67	1932
5	0.68	0.28	0.40	354
micro avg	0.82	0.58	0.68	8752
macro avg	0.69	0.44	0.53	8752
weighted avg	0.81	0.58	0.68	8752
samples avg	0.05	0.05	0.05	8752

for KNeighbors Classifier model..

Accuracy score for Train : 91.27%

Accuracy score for Test : 89.19%

F1 score for Test : 48.65%

Precision for Test : 67.42%

Hamming Loss : 2.94%

Confusion Matrix:

```
[[[35241  797]
 [ 2306 1521]]
```

```
[[[39360  116]
 [   300   89]]
```

```
[[[37480  262]
 [ 1198  925]]
```

```
[[[39711   27]
 [   105   22]]
```

```
[[[37581  352]
 [ 1218  714]]
```

```
[[[39456   55]
 [   295   59]]]
```

Test Classification report:

	precision	recall	f1-score	support
0	0.66	0.40	0.50	3827
1	0.43	0.23	0.30	389
2	0.78	0.44	0.56	2123
3	0.45	0.17	0.25	127
4	0.67	0.37	0.48	1932
5	0.52	0.17	0.25	354
micro avg	0.67	0.38	0.49	8752
macro avg	0.58	0.30	0.39	8752
weighted avg	0.67	0.38	0.48	8752
samples avg	0.03	0.03	0.03	8752

for Random Forest Classifier model..

Accuracy score for Train : 99.10%

Accuracy score for Test : 91.29%

F1 score for Test : 67.88%

Precision for Test : 80.18%

Hamming Loss : 2.04%

Confusion Matrix:

```
[[[35520  518]
 [ 1446 2381]]
```

```
[[[39441   35]
 [   348   41]]
```

```
[[[37464  278]
 [   610 1513]]
```

```
[[[39727   11]
 [   116   11]]
```

```
[[[37532  401]
 [   792 1140]]
```

```
[[[39481   30]
 [   289   65]]]
```

Test Classification report:

	precision	recall	f1-score	support
0	0.82	0.62	0.71	3827
1	0.54	0.11	0.18	389
2	0.84	0.71	0.77	2123
3	0.50	0.09	0.15	127
4	0.74	0.59	0.66	1932
5	0.68	0.18	0.29	354
micro avg	0.80	0.59	0.68	8752
macro avg	0.69	0.38	0.46	8752
weighted avg	0.79	0.59	0.66	8752
samples avg	0.06	0.05	0.05	8752

for StochasticGB Classifier model..

Accuracy score for Train : 91.67%

Accuracy score for Test : 91.47%

F1 score for Test : 62.85%

Precision for Test : 88.73%

Hamming Loss : 2.10%

Confusion Matrix:

```
[[[35891  147]
  [ 1877 1950]]
```

```
[[39476   0]
 [  389   0]]
```

```
[[37615  127]
 [  815 1308]]
```

```
[[39738   0]
 [  127   0]]
```

```
[[37675  258]
 [  967  965]]
```

```
[[39502   9]
 [  318  36]]]
```

Test Classification report:

	precision	recall	f1-score	support
0	0.93	0.51	0.66	3827
1	0.00	0.00	0.00	389
2	0.91	0.62	0.74	2123
3	0.00	0.00	0.00	127
4	0.79	0.50	0.61	1932
5	0.80	0.10	0.18	354
micro avg	0.89	0.49	0.63	8752
macro avg	0.57	0.29	0.36	8752
weighted avg	0.83	0.49	0.61	8752
samples avg	0.05	0.04	0.04	8752

	models_name	accuracies_train	accuracies_test	precision_scores	f1_scores	hamming_losses
5	KNeighbors Classifier	91.27	89.19	67.42	48.65	2.94
1	Multinomial Naive Bayes	91.55	91.26	87.56	60.47	2.21
6	Random Forest Classifier	99.10	91.29	80.18	67.88	2.04
7	StochasticGB Classifier	91.67	91.47	88.73	62.85	2.10
4	LightGBM Classifier	93.10	91.60	81.86	68.20	1.99
2	XGB Classifier	93.54	91.65	83.84	67.12	2.01
3	Linear SVC	92.28	91.74	84.71	67.89	1.96
0	Logistic Regression	92.12	91.76	86.02	66.65	1.99

We selected **Logistic Regressor** for the following reasons:

- Maximum ACCURACY with high Precision value.
- minimum hamming loss.

Key Metrics for success in solving problem under consideration

Following metrics used for evaluation:

```
from sklearn.model_selection import train_test_split, cross_val_score, GridSearchCV

from sklearn.metrics import confusion_matrix, multilabel_confusion_matrix, classification_report, accuracy_score
from sklearn.metrics import roc_auc_score, roc_curve, auc, precision_score, f1_score
from sklearn.metrics import hamming_loss, log_loss
```

Hyperparameter Tuning:

```
1 param = {'estimator__penalty' : ['l1', 'l2', 'elasticnet', 'none'],
2         'estimator__max_iter' : [100, 200, 300, 400],
3         'estimator__multi_class' : ['ovr', 'auto']}
4
5 lgr_ovr = OneVsRestClassifier(lgr)
6 lgr_grid = GridSearchCV(lgr_ovr, param, cv=5, verbose = 2)
7 x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.25, random_state=42)
8 lgr_grid.fit(x_train, y_train)
9 lgr_grid.best_params_
```

```
{'estimator__max_iter': 100,
 'estimator__multi_class': 'ovr',
 'estimator__penalty': 'l2'}
```

```
1 lgr_grid.best_score_
```

```
0.9182658137882018
```

```
1 lgr_tune = LogisticRegression(solver='lbfgs', max_iter= 100, penalty= 'l2', multi_class= 'ovr')
```

```
1 lgr_tune_final = OneVsRestClassifier(lgr_tune)
2 lgr_tune_final.fit(x_train, y_train)
3 pred_train = lgr_tune_final.predict(x_train)
4 pred_test = lgr_tune_final.predict(x_test)
5 print('::::::::::::::::::::::::::::::::::::::::::::::::::::::::::')
6 print(' ')
7 print(' ')
8 print('for ' + models_name[0] + ' model..')
9 print(' ')
10 print(f"Accuracy score for Train : {accuracy_score(y_train, pred_train) * 100:.2f}%")
11 # accuracies_train.append(round(accuracy_score(y_train, pred_train) * 100, 2))
12
13 print(f"Accuracy score for Test : {accuracy_score(y_test, pred_test) * 100:.2f}%")
14 # accuracies_test.append(round(accuracy_score(y_test, pred_test) * 100, 2))
15
16 print(f"F1 score for Test : {f1_score(y_test, pred_test, average='micro') * 100:.2f}%")
17 # f1_scores.append(round(f1_score(y_test, pred_test, average='micro') * 100, 2))
18
19 print(f"Precision for Test : {precision_score(y_test, pred_test, average='micro') * 100:.2f}%")
20 # precision_scores.append(round(precision_score(y_test, pred_test, average='micro') * 100, 2))
21
22 print(f"Hamming Loss : {hamming_loss(y_test, pred_test) * 100:.2f}%")
23 # hamming_losses.append(round(hamming_loss(y_test, pred_test) * 100, 2))
24
25 print(' ')
26 print(' ')
27 print('Confusion Matrix: ')
28 print(' ')
29 print(multilabel_confusion_matrix(y_test, pred_test))
30 print(' ')
31 print(' ')
32 print('Test Classification report: ')
33 print(' ')
34 print(classification_report(y_test, pred_test))
```

for Logistic Regression model..

Accuracy score for Train : 92.12%

Accuracy score for Test : 91.76%

F1 score for Test : 66.65%

Precision for Test : 86.02%

Hamming Loss : 1.99%

Confusion Matrix:

```
[[[35791  247]
  [ 1609 2218]]
```

```
[[39424   52]
  [  306   83]]
```

```
[[37593  149]
  [  776 1347]]
```

```
[[39723   15]
  [  113   14]]
```

```
[[37663  270]
  [  922 1010]]
```

```
[[39470   41]
  [  265   89]]]
```

Test Classification report:

	precision	recall	f1-score	support
0	0.90	0.58	0.71	3827
1	0.61	0.21	0.32	389
2	0.90	0.63	0.74	2123
3	0.48	0.11	0.18	127
4	0.79	0.52	0.63	1932
5	0.68	0.25	0.37	354
micro avg	0.86	0.54	0.67	8752
macro avg	0.73	0.39	0.49	8752
weighted avg	0.85	0.54	0.66	8752
samples avg	0.05	0.05	0.05	8752

Saving & predictions of the model on Test data provided

```
1 filename='Malignant_commet_classifier.pkl'
2 pickle.dump(lgr_tune_final,open(filename,'wb'))
```

```
1 model =pickle.load(open('Malignant_commet_classifier.pkl','rb'))
2 pred =model.predict(fin_test)
```

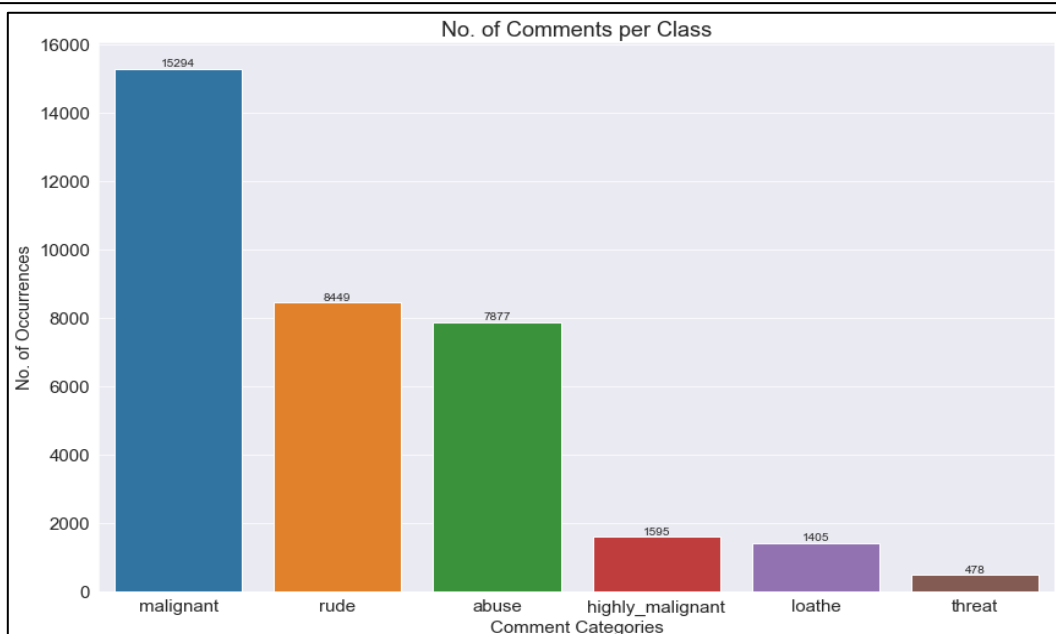
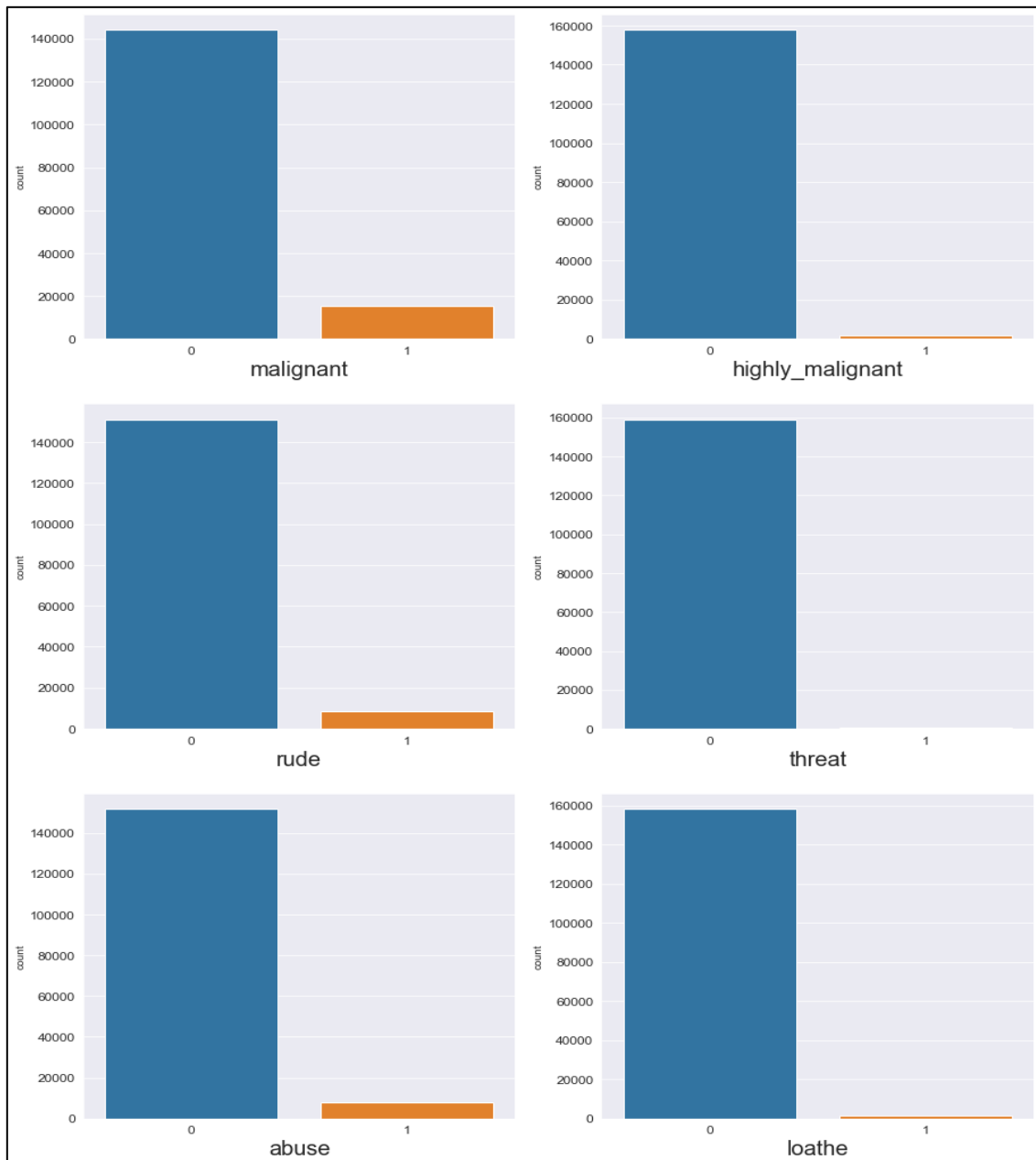
```
1 test_predictions_cleaned = pd.concat([test2['text_lemmatize'],
2                                     pd.DataFrame(pred, columns= ['malignant','highly_malignant','rude',
3                                     'threat','abuse','loathe'])], axis=1)
```

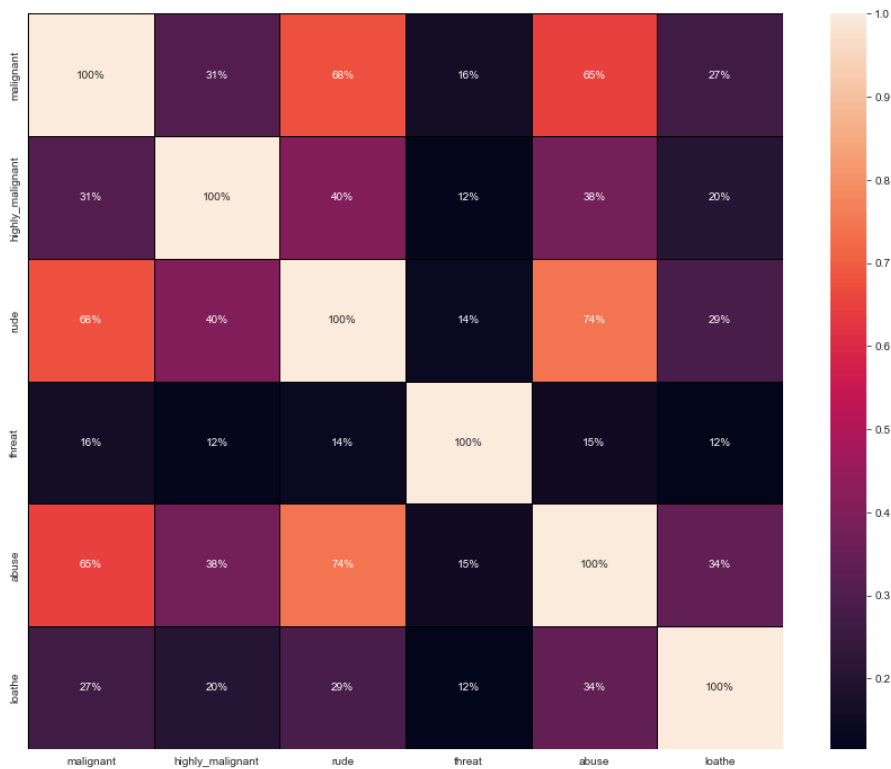
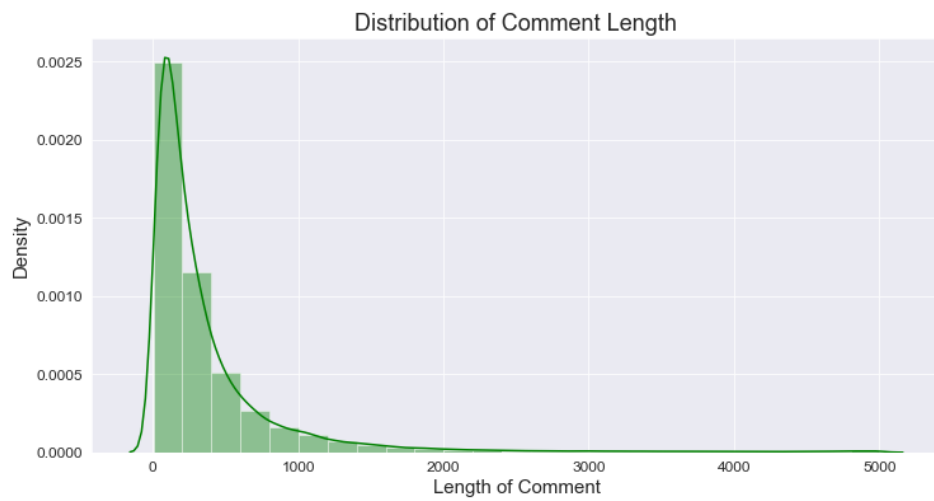
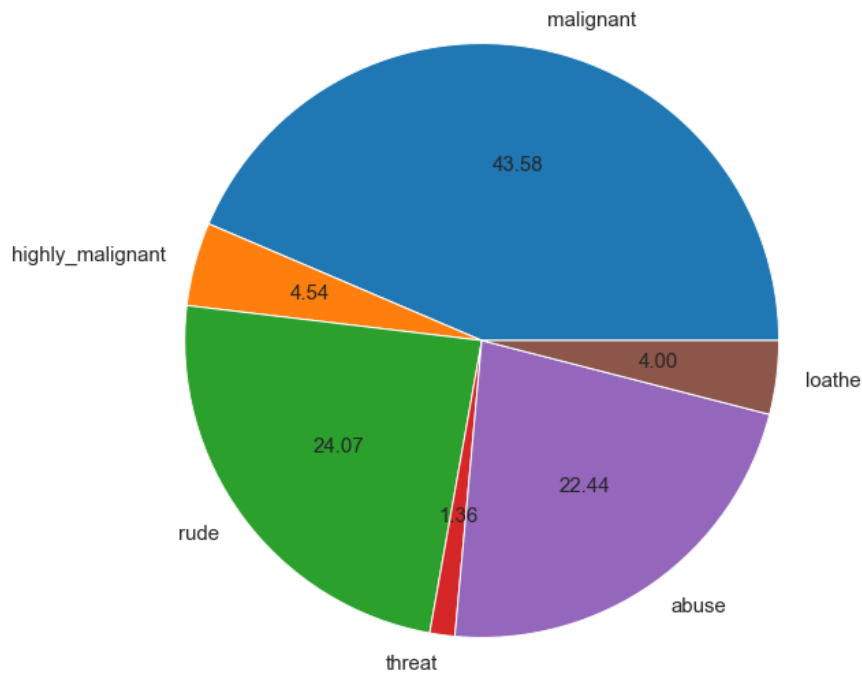
```
1 test_predictions_cleaned.to_csv('test_predictions.csv',index=False)
```

```
1 test_predicted = pd.read_csv('test_predictions.csv')
2 test_predicted
```

	text_lemmatize	malignant	highly_malignant	rude	threat	abuse	loathe
0	yo bitch ja rule succesful ever whats hating s...	1	1	1	0	1	0
1	rfc title fine imo	0	0	0	0	0	0
2	source zawe ashton lapland	0	0	0	0	0	0
3	look back source information updated correct f...	0	0	0	0	0	0
4	anonymously edit article	0	0	0	0	0	0
...
153159	totally agree stuff nothing long crap	1	0	1	0	0	0
153160	throw field home plate get faster throwing cut...	0	0	0	0	0	0
153161	okinotorishima category see change agree corre...	0	0	0	0	0	0
153162	one founding nation eu germany law return quit...	0	0	0	0	0	0
153163	stop already bullshit welcome fool think kind ...	1	0	1	0	1	0

VISUALIZATIONS & EDA





[illegible][illegible]

WORDS TAGGED AS "RUDE"

The word cloud contains the following words: archange, meow, asshole, put, naked, filthy, lolwat, tiger, conversation, libe, got, posting, fuck, peace, work, stupid, Length, dry, great, keep, fuck, edit, drink, deleting, absurd, stop, obese, shark, Name, piss, sit, mother, previous, deleted, fuckeed, cant, object, faggot, text_tokenize, going, stuff, cocksucker, antisemitian, gay, dtype, thought, white, shit, fucking.

[illegible]

[illegible]

Observations from Exploratory Data Analysis:

1. Out of total Negative comments the maximum negative comments come with Malignant in nature followed by rude categories.
2. Around 90% comments are Good/Neutral in nature while rest 10% comments are Negative in nature.
3. Very few comments come with threatening nature.
4. Most of the comments are short with only a few comments longer than 1000 words.
5. Majority of the comments are of length 500, where maximum length is 5000 and minimum is 5.
6. From word cloud, it is clear that negative comments include words like edits, hey, die, bitch, suck, white, crow, absurd, Taliban, fuck/fucking, gay, cocksucker, jew, kill, hate, fagget, piss & shit.

CONCLUSION

Key Findings and Conclusions of the Study

- Most of the comments which are offensive in nature are malignant & these can be classified as having only one label
- Logistic Regressor was best performing model, with accuracy above 90%.

Learning Outcomes of the Study in respect of Data Science

- In this project we learn how to build a machine learning model for classification-based problem.
- The goal of any machine learning problem is to find a single model that will best predict our wanted outcome. Rather than making one model and hoping this model is the best/most accurate predictor we can make, ensemble methods take a myriad of models into account, and average those models to produce one final model.
- So based on all the learning and outcomes our Logistic Regression Model gives the best result so we save this model as our final model for future data/comments predictions.

Limitations of this work and Scope for Future Work

- 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.