MALIGNANT COMMENT CLASSIFICATION PROJECT

Submitted by:

TAMALI SAHA

FlipRobo SME:

GULSHANA CHAUDHARY

ACKNOWLEDGMENT

I would like to express my special gratitude to Flip Robo Technologies team, who has given me this opportunity to deal with this dataset during my internship. It helped me to improve my analyzation skills. I want to express my gratitude to Ms. Gulshana Chaudhary (SME, Flip Robo) as she has helped me to get out of all the difficulties I faced while doing the project. I also want to give huge thanks to entire DataTrained team.

Bibliography:

Reference used in this project:

1. Hands on Machine learning with scikit learn and tensor flow by Aurelien Geron.
2. Andrew Ng Notes on Machine Learning (GitHub).
3. Different projects on Github and Kaggle.
4. Different conference papers on Recharchgate.
5. Toxic Comment Classification by Nupur Baghel.

Table of Contents

1. Introduction

1.1 Business Problem Framing

People can now freely express themselves online due to the growth of social media. However, concurrently, this has led to the emergence of conflict and hatred, making online spaces hostile for users. Despite the fact that researchers have discovered that hate is an issue on a number of platforms, there aren't any models for detecting hate online. Online hatred has been identified as a significant hazard on social media websites, including abusive language, aggressiveness, cyberbullying, hatefulness, and many more. The most common environment for such harmful behaviour is social media platforms.

On various social media platforms, there has been a striking rise in the number of instances of cyberbullying and trolls. People are criticising many celebrities and influencers, and they frequently encounter harsh and abusive remarks. Anyone can suffer from the mental effects of this, which can include despair, mental disease, 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.

1.2 Conceptual Background of the Domain Problem

It has been observed that the number of incidents involving online hatred has skyrocketed in recent years. Nowadays, social media is becoming a hole of darkness and poison for people. Differentiation in viewpoint, race, religion, occupation, nationality, and other factors are the cause of online hatred. People that engage in or disseminate these types of activities on social media sometimes use vulgar language, aggressive behaviour, offensive imagery, and other tactics to insult and seriously harm those on the opposing side. One of the main issues at the moment is this.

Such actions may have dangerous results. The victims experience mental anguish, which makes their lives miserable. For those who are not well-informed about mental health issues, cyberbullying or online hate can be fatal. These cases are also increasing. Religions are also feeling the effects of it. We witness incidents of conflict between members of various communities or religions every single day as a result of inflammatory social media posts.

On-line social media platforms have been highlighted as being particularly vulnerable to online hate, which is defined as abusive language, aggression, cyberbullying, hatefulness, insults, personal assaults, provocation, racism, sexism, threats, or toxicity. For a brighter future, these kinds of behaviours ought to be stopped.

1.3 Review of Literature

Users now leave a lot of comments on various social networks, news websites, and forums. Certain comments are harmful or abusive. Since it is impractical to manually monitor so many comments, the majority of systems employ some form of machine learning models to automatically identify harmful content. In this study, we used machine learning techniques to conduct an in-depth analysis of the state-of-the-art in the classification of toxic comments. First, we looked into the papers' publication dates, locations, and levels of maturity. Each major study's data set, evaluation metric, machine learning techniques, types of toxicity, and comment language were all examined.

1.4 Motivation for the Problem Undertaken

The project is provided to me by Flip Robo Technologies as a part of the internship programme (Internship Batch No-31). This problem is a real world dataset. The exposure of this data gives me the opportunity to locate my skills in solving a real time problem. It is the primary motivation to solve this problem.

We have a lot of possibilities here, but not as many specific solutions. The primary aim is to create a prototype for an online hate and abuse comment classifier that can be used to categorise hateful and offensive comments in order to limit their ability to spread intolerance and cyberbullying.

This study aims to analyse and predicting malignant comment when using **Classification Model** like Logistic Regression, Random Forest, Decision Tree, Gradient Boosting, Extra Tree, Ada Boost Classification algorithms. Thus, the purpose of this study is to grow the knowledge of **Classification methods** in machine learning fields. Those are the different factors to undertaken the problem for study purpose.

2. Analytical Problem Framing

2.1 Mathematical/ Analytical Modelling of the Problem

The goal of this project is to predict malignant comments in social media. We would use a classification method, which is a sort of supervised learning. Although performing a classification seems more logical given that we have between 5 and 6 classes to forecast. We will only do classification in this case. Filtering the words is necessary to avoid overfitting because the dataset only contains one feature.

This project have two big set of data one is training and another is testing. Throughout the project's classification phase, we would first eliminate email addresses, phone numbers, web addresses, spaces, and other terms with stops, in order to calculate the regularisation parameter. We also used TFID to transform the tokens from the train papers into vectors so that the machine could carry out additional processing in order to further enhance our models.

Here 6 different algorithm are used and final model is chosen by best AUC-ROC score and accuracy score.

2.2 Data Sources and their formats

There are two set of data, training and testing. Training dataset has 159571 rows and 8 columns in ot her hand the testing dataset has 153164 rows and 2 columns. The model will train with the help of tra ining dataset. It has 6 integer datatype and 2 object datatype. All integer datatype are actually binary in nature.
Later the training dataset is divided into two parts, training and testing. After determine the proper model, the model is applied to predict the target variable for the test dataset.

```
print('No. of Rows of train dataset :',data.shape[0])
print('No. of Columns of train dataset :',data.shape[1])
```

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

```
data.columns.to_series().groupby(data.dtypes).groups
```

```
{int64: ['malignant', 'highly_malignant', 'rude', 'threat', 'abuse', 'loathe'], object: ['id', 'comment_text']}
```

```
print('No. of Rows of test dataset :',data_test.shape[0])
print('No. of Columns of test dataset :',data_test.shape[1])
```

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

2.3 Data Pre-processing Done:

2.3.1 Feature Engineering:

' --', 'null', 'NA', ' ' are not present in the traing and testing datase.

```
data.isin([' --','null','NA',' ']).sum().any()
```

```
False
```

```
data.isnull().sum()
```

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

```
data_test.isin([' --','null','NA',' ']).sum().any()
```

```
False
```

```
data_test.isnull().sum()
```

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

2.3.2 Drop unnecessary columns:

Let’s drop the unnecessary column ‘id’ from both dataset.

```
data.drop('id',axis=1,inplace=True)
data_test.drop('id',axis=1,inplace=True)
```

2.3.3 Calculate length before cleaning of 'comment_text' column of training and testing dataset:

Let’s calculate the comment length before cleaning.

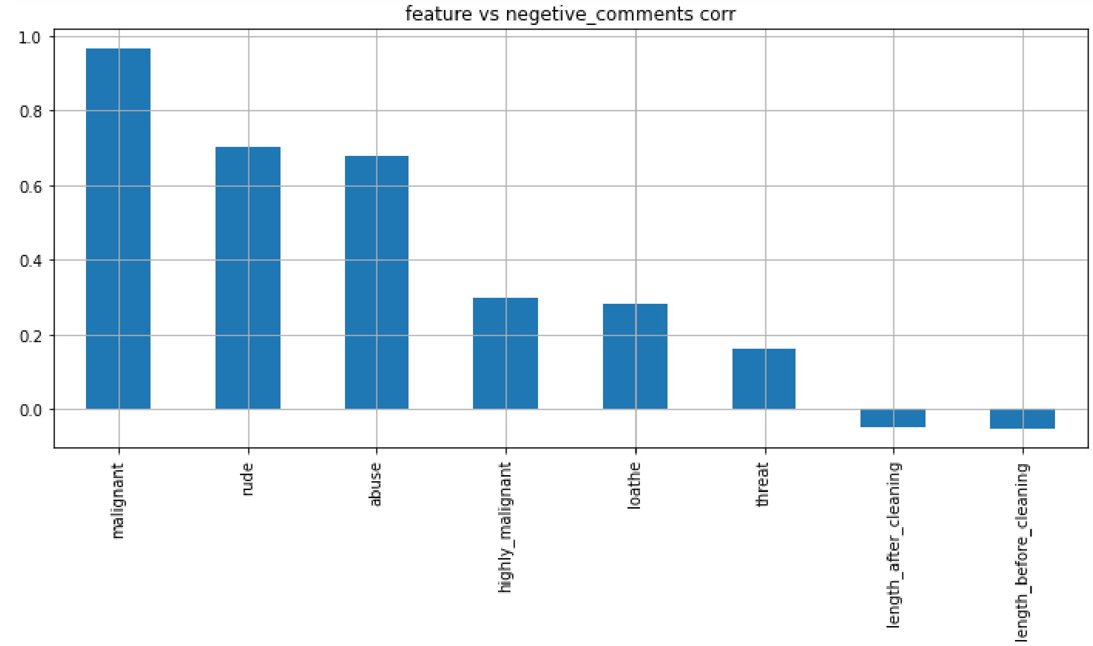
```
data['length_before_cleaning'] = data['comment_text'].str.len()
data.head()
```

	comment_text	malignant	highly_malignant	rude	threat	abuse	loathe	length_before_cleaning
0	Explanation\nWhy the edits made under my usern...	0	0	0	0	0	0	264
1	D'aww! He matches this background colour I'm s...	0	0	0	0	0	0	112
2	Hey man, I'm really not trying to edit war. It...	0	0	0	0	0	0	233
3	"\nMore\nI can't make any real suggestions on ...	0	0	0	0	0	0	622
4	You, sir, are my hero. Any chance you remember...	0	0	0	0	0	0	67

2.3.4 Correlation:

Observations of the correlation:

1. Very obviously length before cleaning and length after cleaning are highly correlated with each other's.
2. Malignant and rude comments are highly correlated with target.
3. The two feature length before cleaning and length after cleaning are negatively correlated with target.



2.3.5 Make new column named negetive_comments:

Here, the comments are 6 different type. If anyone is present the comment tagged as a malignant comment (bad/ negative) comment. So let's make a new column named.

If it is 0= Good comment, 1= malignant comment.

```
target_data = data[['malignant','highly_malignant','rude','threat','abuse','loathe']]

data['negetive_comments'] =data[['malignant','highly_malignant','rude','threat','abuse','loathe']].sum(axis =1)
print(data['negetive_comments'].value_counts())
data['negetive_comments'] = data['negetive_comments'] > 0
data['negetive_comments'] = data['negetive_comments'].astype(int)
print(data['negetive_comments'].value_counts())
```

```
0    143346
1      6360
3      4209
2      3480
4      1760
5       385
6        31
Name: negetive_comments, dtype: int64
0    143346
1     16225
Name: negetive_comments, dtype: int64
```

2.4 Data Inputs- Logic- Output Relationships

We can see in the correlation that every features are correlated with each other and also they are highly correlated with target variable label.

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

No such assumptions are taken for this case.

2.6 Hardware and Software Requirements and Tools Used

Processor: Intel(R) Core(TM) i3-5005U CPU @ 2.00GHz 2.00 GHz
RAM: 4.00 GB
System Type: 64-bit operating system, x64-based processor
Window: Windows 10 Pro
Anaconda – Jupyter Notebook
Libraries Used –

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')

from sklearn.model_selection import train_test_split
```

For Word-Cloud the following libraries are used.

```
#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
```

Except this, different libraries are used for machine learning model building from sklearn.

3. Model/s Development and Evaluation

3.1 Identification of possible problem-solving approaches (methods):

In this problem Classification-based machine learning algorithm like logistic regression can be used. Removed any excess spaces, changed the email addresses subject line to a phone number that is probably wise, etc. For building an appropriate ML model before implementing classification algorithms, data is split in training & test data using train_test_split. Then different statistical parameter like accuracy score, confusion matrix, classification report, precision, recall etc. are determined for every algorithm. Hyper parameter tuning is performed to get the accuracy score much higher and accurate than earlier.

Then the best model is chosen from 6 different algorithm.

3.2 Testing of Identified Approaches (Algorithms)

Total 7 algorithms used for the training and testing are:

1. Logistic Regression
2. Decision Tree Classifier
3. Gradient Boosting Classifier
4. Random Forest Classifier
5. Extra Trees Classifier
6. Ada Boost Classifier

3.3 Key Metrics for success in solving problem under consideration:

From metrics module of sklearn library import classification_report, accuracy_score, confusion_matrix, classification_report and f1_score. From model_selection also, we use cross_val_score. Those are the matrices use to validate the model's quality. Let's discuss every metrics shortly.

- Classification report: It is a performance evaluation metric in machine learning which is used to show the precision, recall, F1 Score, and support score of your trained classification model
- Accuracy score: It is used when the True Positives and True negatives are more important. Accuracy can be used when the class distribution is similar.
- Confusion Matrix: It is a table that is used in classification problems to assess where errors in the model were made. The rows represent the actual classes the outcomes should have been. While the columns represent the predictions we have made. Using this table it is easy to see which predictions are wrong.
- Precision: It can be seen as a measure of quality. If the precision is high, an algorithm returns more relevant results than irrelevant ones.
- Recall: The recall is calculated as the ratio between the numbers of Positive samples correctly classified as Positive to the total number of Positive samples.
- F1 Score: F1 = 2 * (precision * recall) / (precision + recall)

3.4 Run and Evaluate selected models

First find the best random state of train_test_split to get best accuracy. Here the random state is 88. Then after splitting the data into 4 different part and check the shape of the data.

```
print('Training feature shape:',x_train.shape)
print('Training target shape:',y_train.shape)
print('Test feature shape:',x_test.shape)
print('Test target shape:',y_test.shape)
```

```
Training feature shape: (119678, 51153)
Training target shape: (119678,)
Test feature shape: (39893, 51153)
Test target shape: (39893,)
```

A Logistic Regression:

```
from sklearn.linear_model import LogisticRegression
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size = 0.25, random_state= 88)
log = LogisticRegression()
log.fit(x_train, y_train)
y_pred = log.predict(x_test)

print('accu score : ', accuracy_score(y_test, y_pred))
print ('cof_mat:\n ', confusion_matrix(y_test, y_pred))
print('classification report:\n ', classification_report(y_test, y_pred))
print("-----------")
print("-----------")
print('training score : ', log.score(x_train, y_train))
print('testing score : ', log.score(x_test, y_test))
```

```
accu score :  0.957762013385807
cof_mat:
  [[35763   144]
 [ 1541  2445]]
classification report:
               precision    recall  f1-score   support

           0       0.96      1.00      0.98     35907
           1       0.94      0.61      0.74      3986

    accuracy                           0.96     39893
   macro avg       0.95      0.80      0.86     39893
weighted avg       0.96      0.96      0.95     39893

-----------
-----------
training score :  0.9591236484566921
testing score :  0.957762013385807
```

In this way accuracy score is determined for each 6 different classification model.

B Decision Tree Classifier:

The accuracy score, confusion matrix and classification report after using Decision tree Classifier is as follows.

```
accu score :  0.9453287544180683
cof_mat:  [[34822  1085]
 [ 1096  2890]]
classification report:

              precision    recall  f1-score   support

           0       0.97      0.97      0.97     35907
           1       0.73      0.73      0.73      3986

    accuracy                           0.95     39893
   macro avg       0.85      0.85      0.85     39893
weighted avg       0.95      0.95      0.95     39893

-----------
-----------
training score :  0.9994735874596835
testing score :  0.9453287544180683
```

C Gradient Boosting Classifier:

The accuracy score, confusion matrix and classification report after using Gradient Boosting Classifier is as follows.

```
accu score :  0.9420449702955406
cof_mat:  [[35816    91]
 [ 2221  1765]]
classification report:

              precision    recall  f1-score   support

           0       0.94      1.00      0.97     35907
           1       0.95      0.44      0.60      3986

    accuracy                           0.94     39893
   macro avg       0.95      0.72      0.79     39893
weighted avg       0.94      0.94      0.93     39893

-----------
-----------
training score :  0.9393372215444776
testing score :  0.9420449702955406
```

D Random Forest Classifier:

The accuracy score, confusion matrix and classification report after using Random Forest Classifier is as follows.

```
accu score :  0.9588148296693656
cof_mat:  [[35610   297]
 [ 1346  2640]]
classification report:
              precision    recall  f1-score   support
           0       0.96      0.99      0.98     35907
           1       0.90      0.66      0.76      3986

    accuracy                           0.96     39893
   macro avg       0.93      0.83      0.87     39893
weighted avg       0.96      0.96      0.96     39893

-----------
-----------
training score :  0.9994568759504671
testing score :  0.9588148296693656
```

E Extra Trees Classifier:

The accuracy score, confusion matrix and classification report after using Extra Trees Classifier is as follows.

```
accu score :  0.9586142932344021
cof_mat:  [[35545   362]
 [ 1289  2697]]
classification report:
              precision    recall  f1-score   support

           0       0.97      0.99      0.98     35907
           1       0.88      0.68      0.77      3986

    accuracy                           0.96     39893
   macro avg       0.92      0.83      0.87     39893
weighted avg       0.96      0.96      0.96     39893
-----------
-----------
training score :  0.9994735874596835
testing score :  0.9586142932344021
```

F [Ada Boost Classifier:]

The accuracy score, confusion matrix and classification report after using Ada Boost Classifier is as follows.

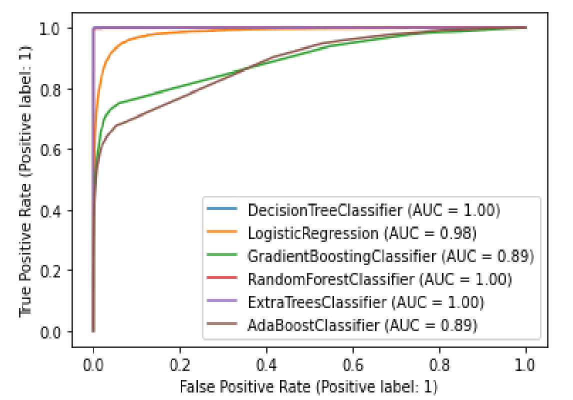
```
accu score :  0.9477101245832602
cof_mat:  [[35629   278]
 [ 1808  2178]]
classification report:
               precision    recall  f1-score   support

           0       0.95      0.99      0.97     35907
           1       0.89      0.55      0.68      3986

    accuracy                           0.95     39893
   macro avg       0.92      0.77      0.82     39893
weighted avg       0.95      0.95      0.94     39893

-----------
-----------
training score :  0.944818596567456
testing score :  0.9477101245832602
```

As per 6 different model, for Random Forest the accuracy score is highest among all the models. But the difference between training and testing is large.

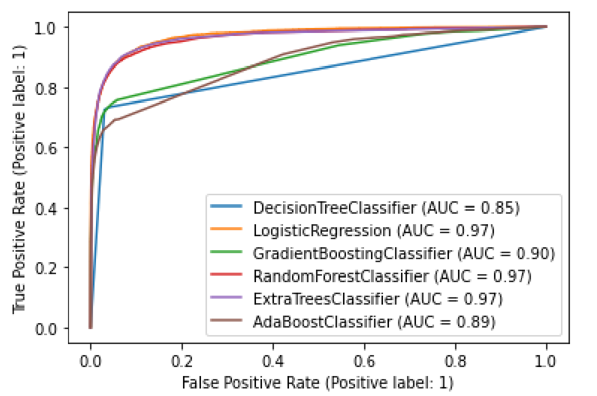
But for Logistic Regression the difference between training and testing is very small as well as it also gives good accuracy.

3.5 AUC- ROC Curve:

```
from sklearn.metrics import plot_roc_curve

disp = plot_roc_curve(clf, x_train, y_train)

plot_roc_curve(log, x_train, y_train, ax=disp.ax_)
plot_roc_curve(gbdt, x_train, y_train, ax=disp.ax_)
plot_roc_curve(rf, x_train, y_train, ax=disp.ax_)
plot_roc_curve(etc, x_train, y_train, ax=disp.ax_)
plot_roc_curve(ada, x_train, y_train, ax=disp.ax_)

plt.show()
```




```
from sklearn.metrics import plot_roc_curve

disp = plot_roc_curve(clf, x_test, y_test)

plot_roc_curve(log, x_test, y_test, ax=disp.ax_)
plot_roc_curve(gbdt, x_test, y_test, ax=disp.ax_)
plot_roc_curve(rf,x_test, y_test, ax=disp.ax_)
plot_roc_curve(etc, x_test, y_test, ax=disp.ax_)
plot_roc_curve(ada, x_test, y_test, ax=disp.ax_)

plt.show()
```



3.6 Hyper Parameter Tuning:

Here for Random Forest, Logistic Regression and Extra tree classifier, the AUC score is same. Here also we take Logistic Regression for final model. So it is the final model for this dataset.

```
from sklearn.model_selection import GridSearchCV
grid = dict(solver=['newton-cg', 'lbfgs', 'liblinear'],penalty=['l2','l1'], C=[1.0, 0.1,0.01])

grid_log = GridSearchCV(estimator=log, param_grid= grid,refit = True, verbose = 3 )

grid_log.fit(x_train, y_train)
print('best params : ', grid_log.best_params_)
```

Best params: {'C': 1.0, 'penalty': 'l1', 'solver': 'liblinear'}

Here accuracy score is slightly improved after using hyper parameter tuning. First, accuracy score was 0.957762013385807, but after applying hyper parameter tuning it is 0.9624495525530795.

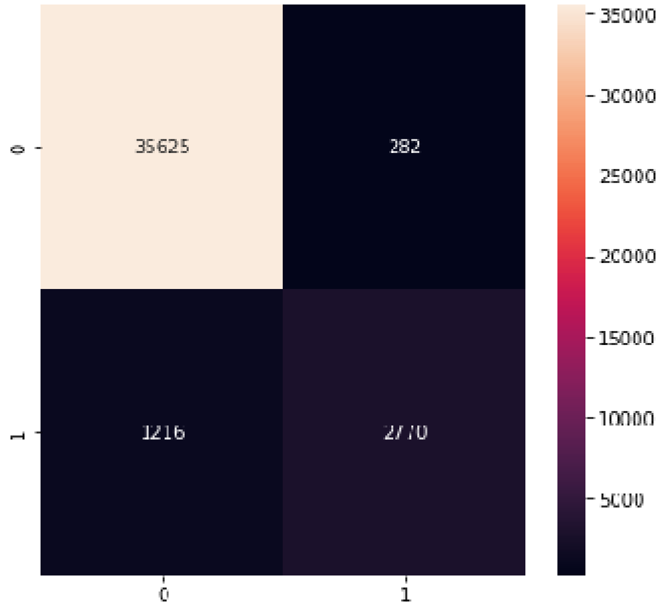
3.7 Final Model:

For final model the target variable is as follows after using Logistic Regression.

```
accu score :  0.9624495525530795
cof_mat:
  [[35625   282]
 [ 1216  2770]]
classification report:
                precision    recall  f1-score   support

           0       0.97      0.99      0.98     35907
           1       0.91      0.69      0.79      3986

    accuracy                           0.96     39893
   macro avg       0.94      0.84      0.88     39893
weighted avg       0.96      0.96      0.96     39893

-----------
-----------
log loss: 1.2969521601797698
training score :  0.962482661809188
testing score :  0.9624495525530795
```

3.8 Confusion Matrix:

```
conf = confusion_matrix (y_test, y_pred)
fig , ax = plt.subplots(figsize=(6,6))
sns.heatmap(conf, annot = True, fmt = ".0f")
plt.show()
```



3.9 Load the model:

Let’s save the model using pickle for future use. Then see the actual and predicted value of 6 random sample.

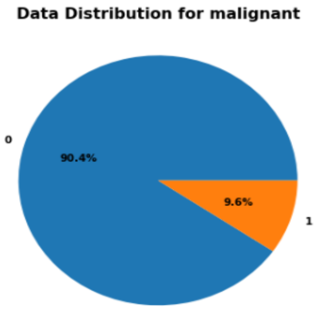
```
import pickle
pickle.dump(grid_log_best, open("Malgnant_Classification_model", "wb"))
load_Malignant_Classification_model= pickle.load(open("Malgnant_Classification_model", "rb"))
```

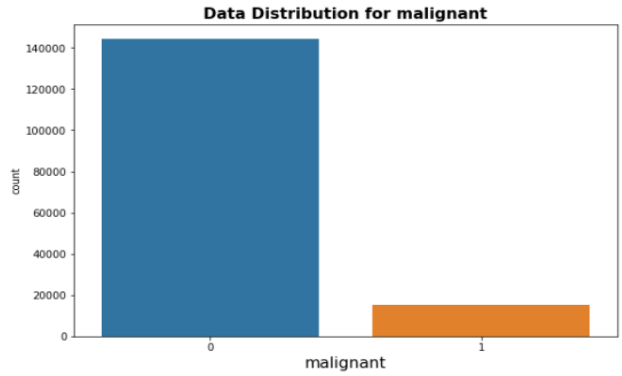
```
y_pred = load_Malignant_Classification_model.predict(x_test)

y_test = np.array(y_test)
data_prediction_by_model = pd.DataFrame()
data_prediction_by_model["Predicted Values"] = y_pred
data_prediction_by_model["Actual Values"] = y_test
data_prediction_by_model.sample(n=6)
```

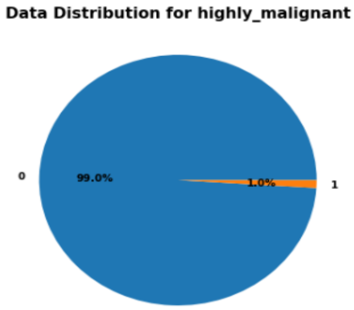
	Predicted Values	Actual Values
36630	0	0
810	0	0
30106	0	0
10115	1	1
18325	0	0
34939	0	0

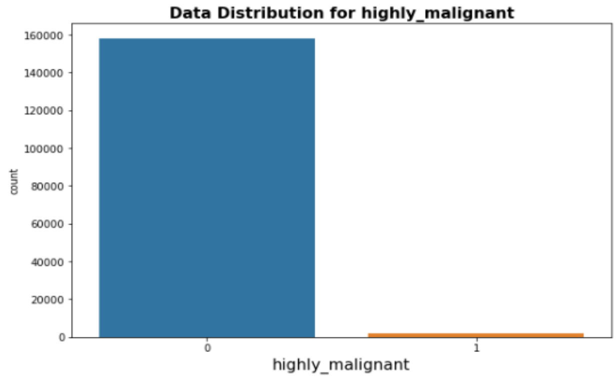
3.10 Visualizations:

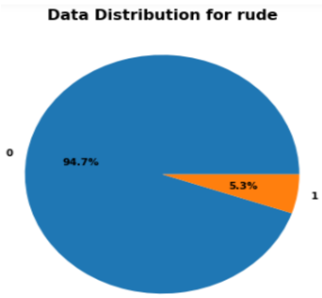
Let's start the observation exploration of feature analysis.

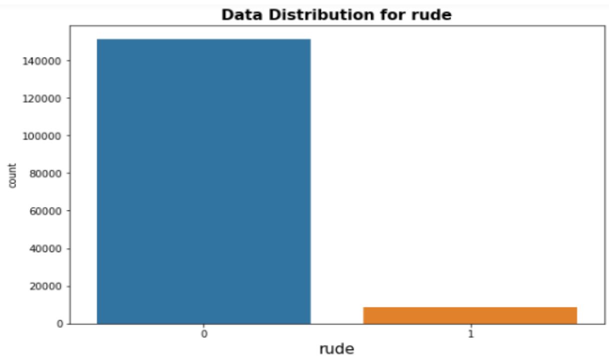






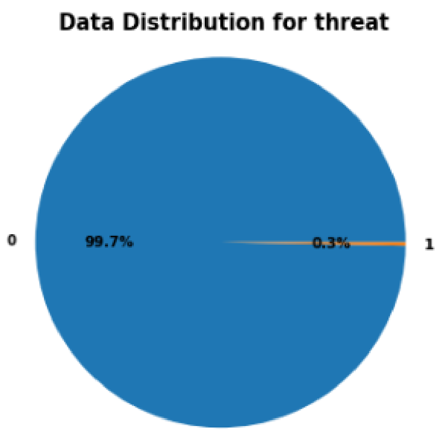


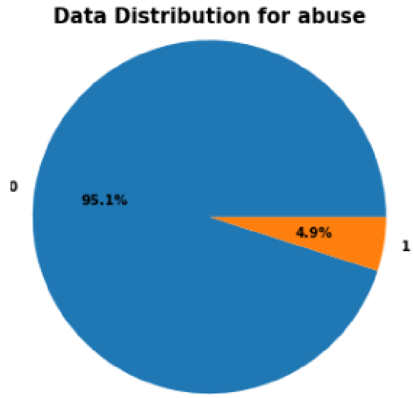


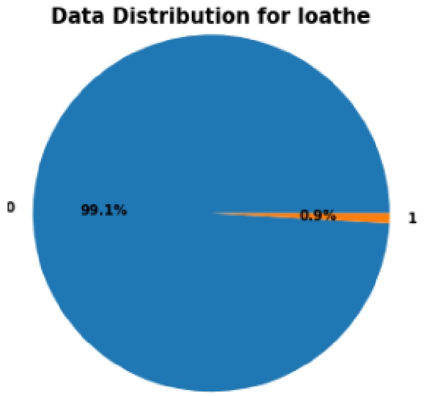


Observations:

1. For malignant comment distribution around 10% comment is malignant while 90% are good comments.
2. For highly_malignant comment distribution around 1% comment is highly_malignant while 99% are good comments.
3. For rude comment distribution around 5% comment is rude while 95% are good comments.

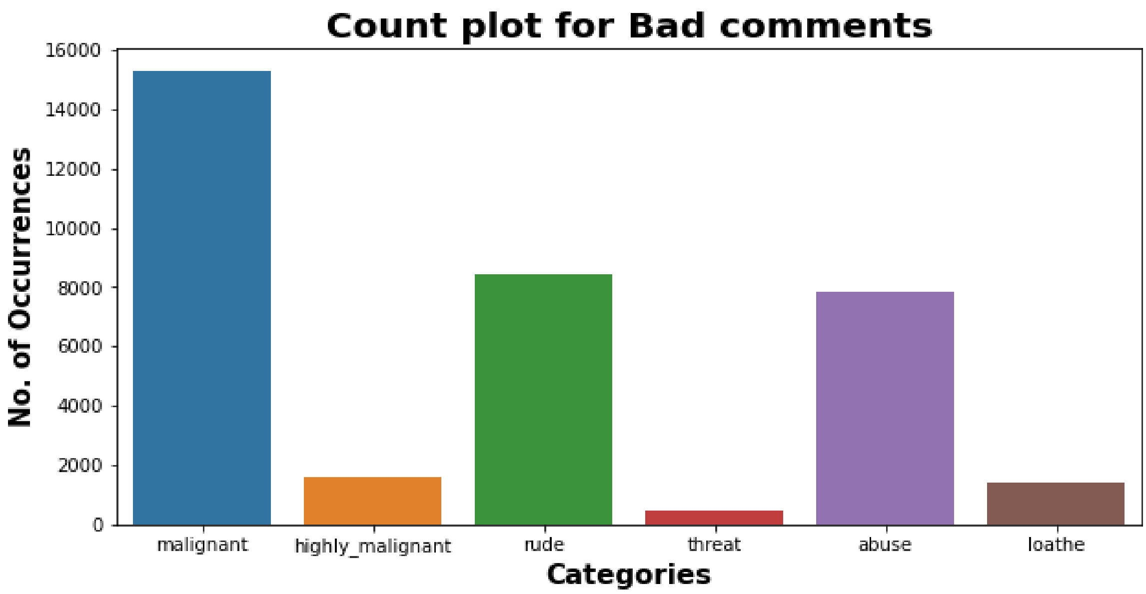


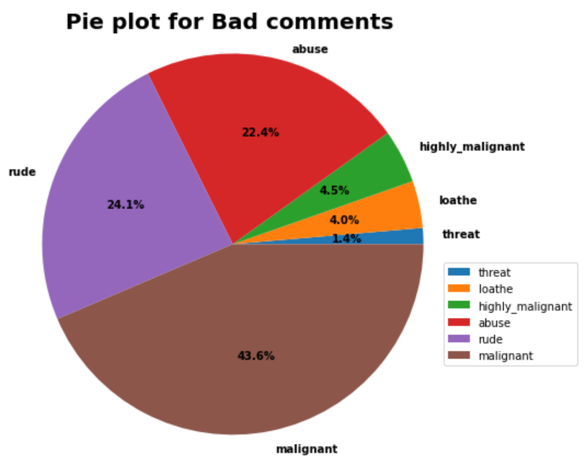




Observations:

1. For threat comment distribution only 0.3% comment is highly_malignant while 99.7% are not threatening comments.
2. For abuse comment distribution around 5% comment is rude while 95% are good comments.
3. For loathe comment distribution around 1% comment is rude while 99% are not loathe comments.



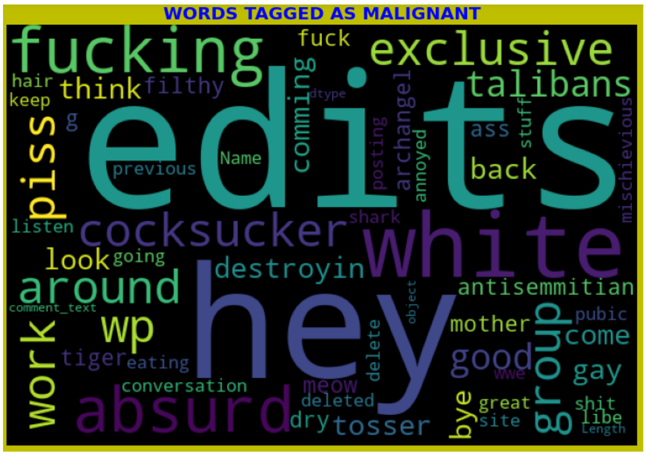


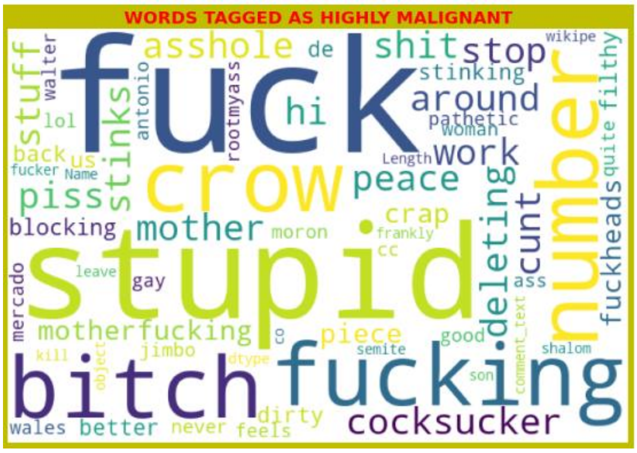
Observations:

1. The maximum negative comments comes with Malignant in nature followed by rude categories.
2. Very few comments comes with threatening nature.
3. Total percentage of negative comment is 10.2% while good comment is 89.8%.
4. Around 90% comments are good while rest 10% comments are Negative in nature.
5. Out of total negative comments around 43.58% are malignant in nature followed by 24.07% are rude comments.

3.11 Word Cloud for different Feature:

We can see from the word clouds above that small texts are given less weight in their respective comment types than large texts are.

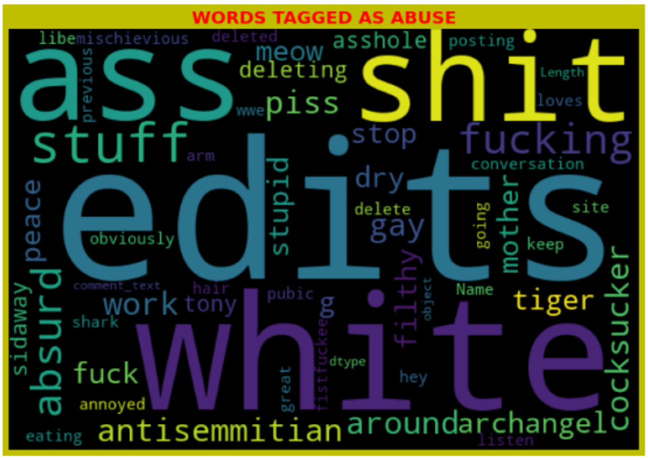




Observations:

1. From word cloud of malignant comments, it is clear that it mostly consists of words like edits, hey, white, fucking, absurd, piss, cocksucker, Taliban etc.
2. From word cloud of highly malignant comments, it is clear that it mostly consists of words like fuck, stupid, fucking, stupid, cocksucker, crow, piss, bitch, around, asshole etc.

3. From word cloud of rude comments, it is clear that it mostly consists of words like shit, fucking, stuff, fucked, white, absurd, piece etc.
4. From word cloud of threat comments, it is clear that it mostly consists of words like die, bitch, fuck, suck, stupid, back, hey, hi, back, last etc.



Observations:

1. From word cloud of abuse comments, it is clear that it mostly consists of words like edits, white, ass, stuff, shit, piss, fucking, cocksucker, antisemmitian, gay etc.
2. From word cloud of abuse comments, it is clear that it mostly consists of words like fuck, gay, jew, kill, antisemmitian, think etc.

3.12 Interpretation of the Results

After all the pre-processing steps, the dataset is ready to train machine learning models. All unnecessary words from comment text are deleted as they might give overfitting problem as well as it also could increase the time complexity. Now apply this dataset on different ML Classification Model (as discussed on part 3.4 - ‘Run and Evaluate selected models’) and check the best model for this particular dataset.

4. CONCLUSION

4.1 Key Findings and Conclusions of the Study

Here, we observed the various detrimental effects that toxic or damaging social media remarks have on society. The ability to quickly and effectively identify remarks as hazardous could have a wide range of positive effects while also reducing the negative ones. We have also seen how easily accessible algorithms can be used in this way to handle this difficulty. It was shown in our particular investigation that a logistic regression solution offers a significant improvement in classification compared to any other approach.

4.2 Learning Outcomes of the Study in respect of Data Science

Data cleansing is one of the most crucial phases; I attempted to make comments shorter and included all the relevant keywords in it. The power of visualisation is beneficial for converting data into a graphical representation; it helps me to comprehend what the data is trying to communicate.

In this dataset, I utilised a variety of methods to find the best result and preserve that model. Logistic regression is the most effective algorithm.

4.3 Limitations of this work and Scope for Future Work

Additionally, the following studies are examples that might be taken into account for future work in this field:

- We offer the following strategy to enhance NLP classifiers: Convolutional neural networks (CNN) and Support vector clustering (SVC) are two additional algorithms that can be used to enhance the performance of existing classifiers. In the present study, the issue was reduced to two classes, although it is worthwhile to pursue the primary objective of six classes of remarks.
- For text processing and text classification, we also advocate the use of SVM. To achieve the best results, a grid search is necessary for hyper-parameter optimization.