MACHINE LEARNING APPLIED: MALAIGNANT COMMENTS CLASSIFICATION.

Submitted by:

MD Hamdan H

ACKNOWLEDGMENT

Foremost, I would like to express my sincere gratitude to Data Trained team for the continuous support of my Data Science study and research, for the patience, motivation, enthusiasm, and immense knowledge. The guidance helped me in all the time of research and writing of this thesis. I could not have imagined having a better advisor and mentor for my Data science study.

Besides Data Trained, I would like to thank Flip Robo Team, for their encouragement, insightful internship, and help to understand the study.

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, 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.

Conceptual Background of the Domain Problem

Online platforms and social media become the place where people share the thoughts freely without any partiality and overcoming all the race people share their thoughts and ideas among the crowd.

Social media is a computer-based technology that facilitates the sharing of ideas, thoughts, and information through the building of virtual networks and communities. By design, social media is Internet-based and gives users quick electronic communication of content. Content includes personal information, documents, videos, and photos. Users engage with social media via a computer, tablet, or smartphone via web-based software or applications.

While social media is ubiquitous in America and Europe, Asian countries like India lead the list of social media usage. More than 3.8 billion people use social media.

In this huge online platform or an online community there are some people or some motivated mob wilfully bully others to make them not to share their thought in rightful way. They bully others in a foul language which among the civilized society is seen as ignominy. And when innocent individuals are being bullied by these mob these individuals are going silent without speaking anything. So, ideally the motive of this disgraceful mob is achieved.

To solve this problem, we are now building a model that identifies all the foul language and foul words, using which the online platforms like social media principally stops these mob using the foul language in an online community or even block them or block them from using this foul language.

Review of Literature

The purpose of the literature review is to:

1. Identify the foul words or foul statements that are being used.
2. Stop the people from using these foul languages in online public forum.

To solve this problem, we are now building a model using our machine language technique that identifies all the foul language and foul words, using which the online platforms like social media principally stops these mob using the foul language in an online community or even block them or block them from using this foul language.

I have used 5 different Classification algorithms and shortlisted the best on basis on the metrics of performance and I have chosen one algorithm and build a model in that algorithm.

Motivation for the Problem Undertaken

One of the first lessons we learn as children is that the louder you scream and the bigger of a tantrum you throw, you more you get your way. Part of growing up and maturing into an adult and functioning member of society is learning how to use language and reasoning skills to communicate our beliefs and respectfully disagree with others, using evidence and persuasiveness to try and bring them over to our way of thinking. Social media is reverting us back to those animalistic tantrums, schoolyard taunts and unfettered bullying that define youth, creating a dystopia where even renowned academics and dispassionate journalists transform from Dr. Jekyll into raving Mr. Hydes, raising the critical question of whether social media should simply enact a blanket ban on profanity and name calling? Actually, ban should be implemented on these profanities and taking that as a motivation I have started this project to identify the malignant comments in social media or in online public forms.

Analytical Problem Framing

Mathematical/ Analytical Modeling of the Problem

I start analysis on this project in importing the data set and simple play around with the data and identifying the characteristics of each column.

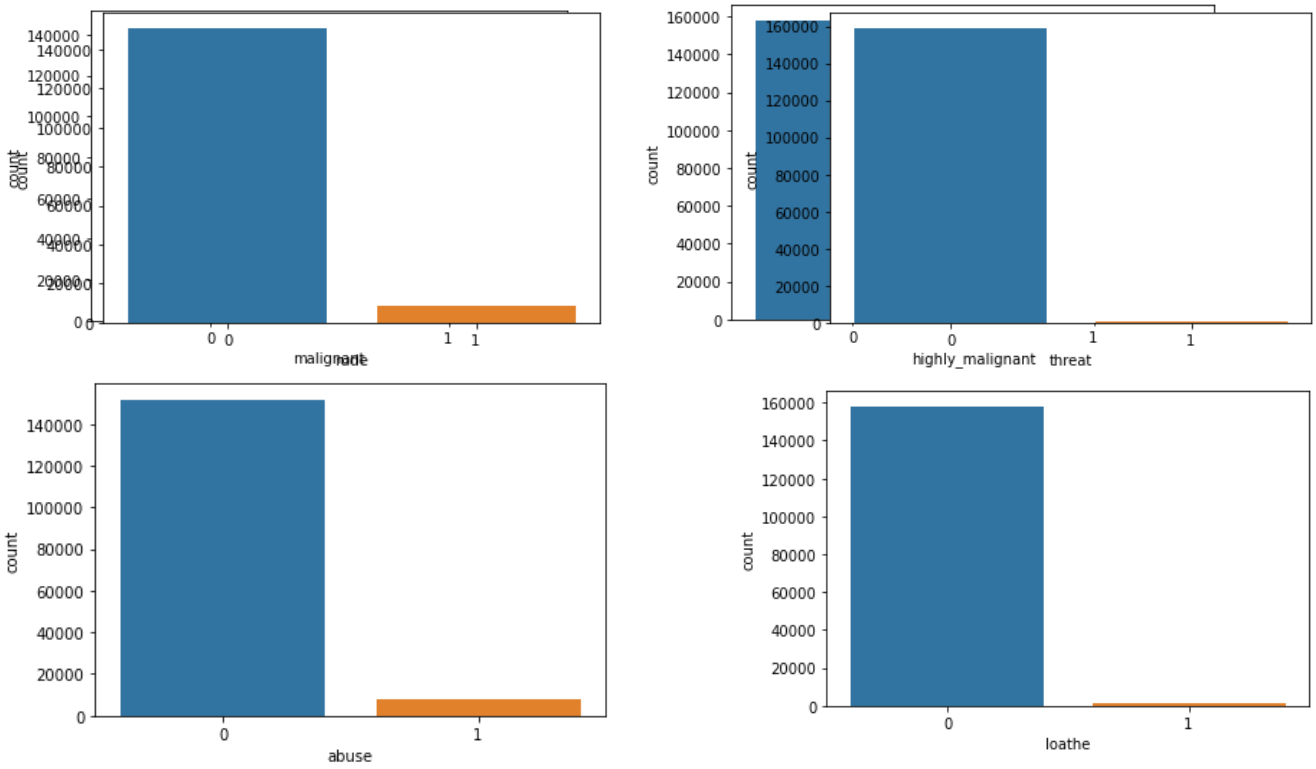
```
In [11]: df_train.dtypes

Out[11]: comment_text       object
         malignant           int64
         highly_malignant    int64
         rude                int64
         threat              int64
         abuse               int64
         loathe              int64
         dtype: object
```

In the first stoke of analysis I understood that there are 8 columns in total in which 6 are numerical column with binary data 0's and 1's and 'id' data which has all unique values "connect text" have string values.

Since 'id' have all unique values, it won't be helpful in analysis to I have dropped id column.

```
#Id in the data wont be of much use we are droping the same
df_train.drop('id',axis=1,inplace=True)
```

Post dropping 'id' I tried to understand data from 'malignant', 'highly_malignant', 'rude', 'threat', 'abuse', 'loathe columns by plotting them in count plot.



Key observation:

we can see that there only minimum number of columns in 'malignant', 'highly_malignant', 'rude', 'threat', 'abuse', 'loathe' and remaining all in 0.

```
#As we visualize above the good coment is much higher than the bad comments lets see that mathematicaly.
Good_comment = df_train[(df_train['malignant']!=1) & (df_train['highly_malignant']!=1) & (df_train['rude']!=1) &
                        (df_train['threat']!=1) & (df_train['abuse']!=1) & (df_train['loathe']!=1)]
percent=len(Good_comment)/len(df_train)*100
print('Percentage of good/neutral comments = ',percent)
print('Percentage of negative comments = ', (100-percent))
```

```
Percentage of good/neutral comments =  89.83211235124176
Percentage of negative comments =  10.167887648758239
```

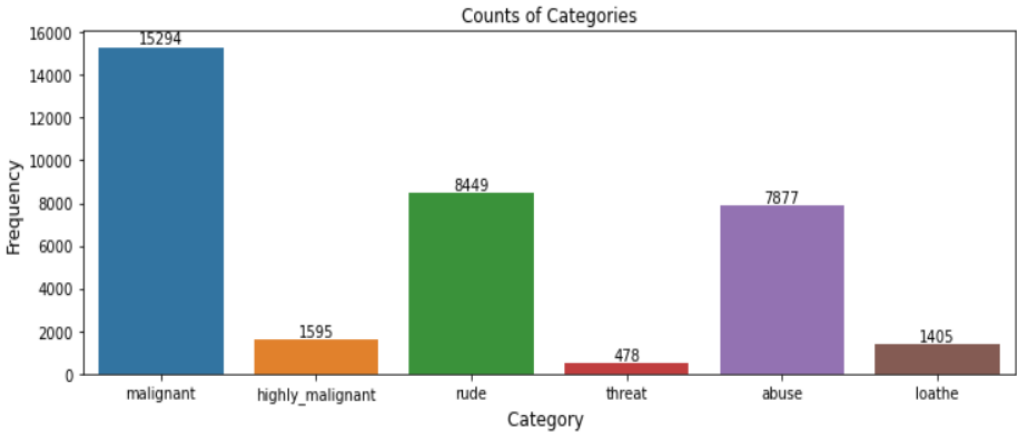
we only of 10.167% of bad comments in the data

So, only 10% of the data is getting classified as malignant comments data is unbalanced.

```
#Storing the number of counts for every target label
counts=df_train.iloc[:,1:].sum()
counts
```

```
malignant          15294
highly_malignant    1595
rude                8449
threat               478
abuse               7877
loathe              1405
dtype: int64
```

```
#Plotting the counts of each category
plt.figure(figsize=(12,4))
ax = sns.barplot(counts.index, counts.values)
plt.title("Counts of Categories")
plt.ylabel('Frequency', fontsize=12)
plt.xlabel('Category ', fontsize=12)
rects = ax.patches
labels = counts.values
for rect, label in zip(rects, labels):
    height = rect.get_height()
    ax.text(rect.get_x() + rect.get_width()/2, height + 5, label, ha='center', va='bottom')
plt.show()
```

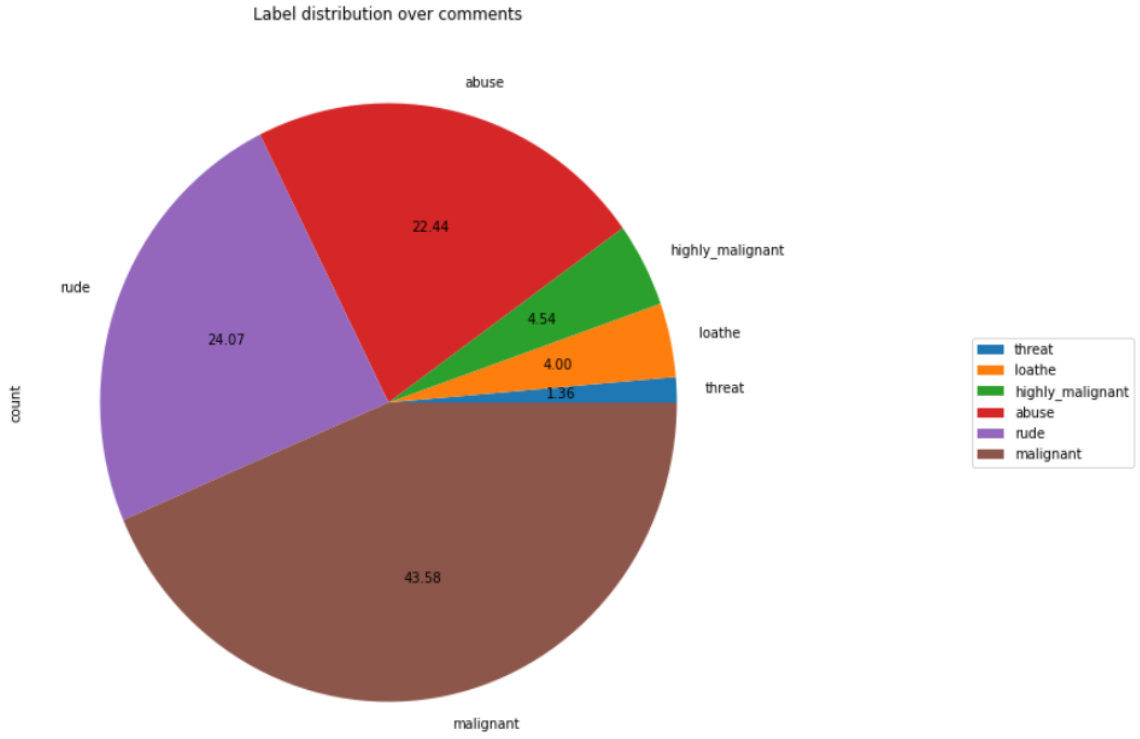


Key observation:

1. We can see malignant and rude are high categorised sentences in the data.

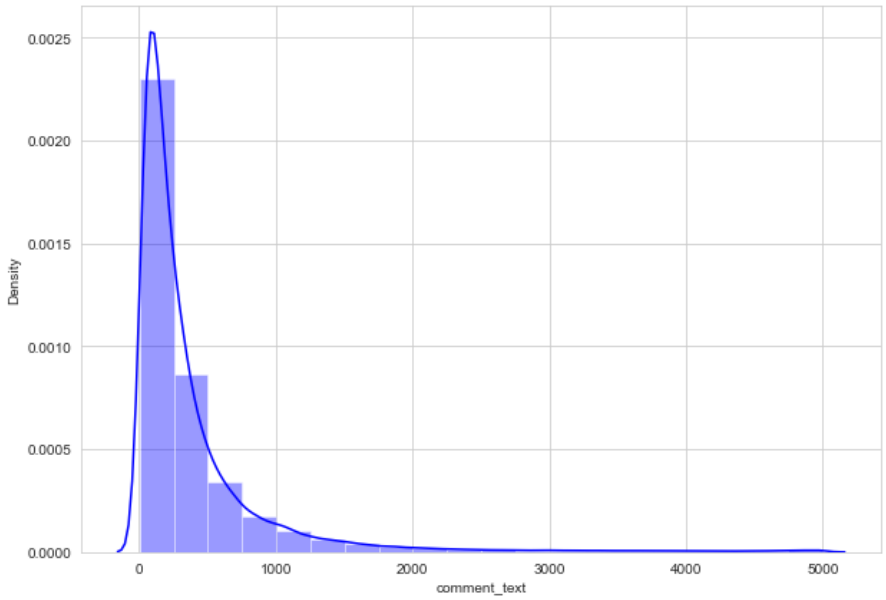
```
df_distribution.plot.pie(y = 'count', title = 'Label distribution over comments', autopct='%.2f', figsize = (20, 10))\
                        .legend(loc='center left', bbox_to_anchor=(1.3, 0.5))
```

```
<matplotlib.legend.Legend at 0x1bbb552b2b0>
```



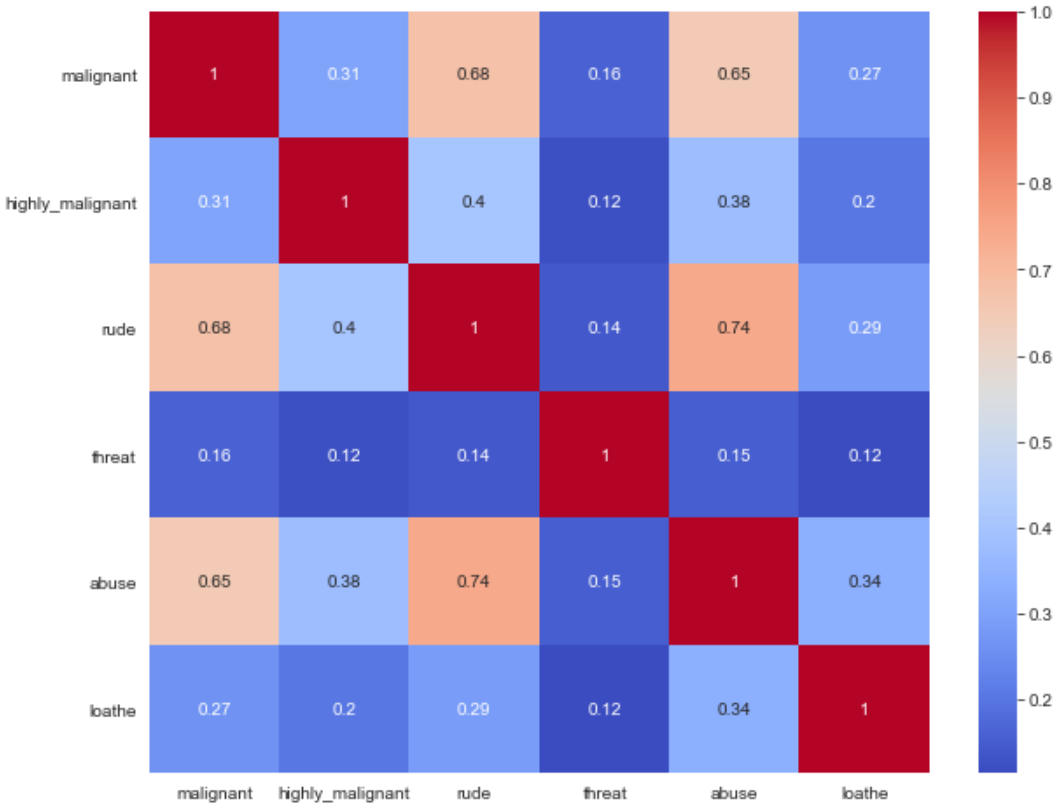
Key observation:

1. As we see above malignant, and rude sentence are high classified and threat, loathe are least classified.



Key observation:

1. We can see that few sentences are really long but most of the sentence are small.



Key Observations:

1. We can see more corelations in the variables, Abuse have more corelation with malignant and rude.
2. Rude has more positive corelation with malignant
3. we don't have any negative corelations in the data.

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:

1. Malignant: It is the Label column, which includes values 0 and 1, denoting if the comment is malignant or not.
2. Highly Malignant: It denotes comments that are highly malignant and hurtful.
3. Rude: It denotes comments that are very rude and offensive.
4. Threat: It contains indication of the comments that are giving any threat to someone.
5. Abuse: It is for comments that are abusive in nature.
6. Loathe: It describes the comments which are hateful and loathing in nature.
7. ID: It includes unique Ids associated with each comment text given.
8. 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.

Data Pre-processing Done

I imported all the required libraries for cleansing the data.

```
#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
```

After importing all the required libraries, I have defined stopwords and lemmatize to a variable.

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

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

Post which I have defined a function for cleaning the data.

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

#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)
```

Post on creating a function I have passed my data into the same to clean it.

```
#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']
```

```
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
                              ...
159566    second time asking view completely contradicts...
159567                 ashamed horrible thing put talk page
159568    spitzer umm there actual article prostitution ...
159569    look like actually put speedy first version de...
159570    really think understand came idea bad right aw...
Name: comment_text, Length: 159571, dtype: object
```

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

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

The total amount of data that is cleansed from the original data is 24418290. Now the data is cleansed and ready for training but before which I converted the data into vectors for the machine learning models to understand the data, so I imported TFIDF vectorizer and I have made the max feature as 15000.

```
#Converting the features into number vectors
tf_vec = TfidfVectorizer(max_features = 15000, 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'])
```

```
y =df_train['label']
```

And I have split the data into two parts X and y and made them ready for training. I have created a new feature name label and summed all the other numerical column and changed the output as binary i.e., if the sentence is categorised as malignant it will be 1 or else it will be 0.

Data Inputs- Logic- Output Relationships

I have analysed the input output logic with word cloud and I have word clouded the sentenced that as classified as foul language in every category.

Key observation.

We can see the foul words that are mostly used in malignant classified sentences we are seeing top 400 words the words which are bigger in size are mostly used.

WORDS TAGGED AS HIGHLY MALIGNANT

Key observation.

We can see the foul words that are mostly used in highly_malignant classified sentences we are seeing top 400 words the words which are bigger in size are mostly used.

WORDS TAGGED AS RUDE

Key observation.

We can see the foul words that are mostly used in rude classified sentences we are seeing top 400 words the words which are bigger in size are mostly used.

WORDS TAGGED AS ABUSE

Key observation.

We can see the foul words that are mostly used in abuse classified sentences we are seeing top 400 words the words which are bigger in size are mostly used.

WORDS TAGGED AS LOATHE

Key observation.

We can see the foul words that are mostly used in loathe classified sentences we are seeing top 400 words the words which are bigger in size are mostly used.

Hardware and Software Requirements and Tools Used

1. Python 3.8.
2. NumPy.
3. Pandas.
4. Matplotlib.
5. Seaborn. 6. Data science.
6. SciPy
7. Sklearn.
8. Anaconda Environment, Jupyter Notebook.

Model/s Development and Evaluation

Testing of Identified Approaches (Algorithms)

I have started the training in selecting the best random state parameter for the model as follows.

Training the model

```
### Selecting parameters for training
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
from sklearn.model_selection import train_test_split

accu = 0
for i in range(0,500):
    x_train, x_test, y_train, y_test = train_test_split(X,y,test_size = .25, random_state = i)
    mod = LogisticRegression()
    mod.fit(x_train,y_train)
    y_pred = mod.predict(x_test)
    acc = accuracy_score(y_test,y_pred)
    if acc> accu:
        accu= acc
        best_rstate=i

print(f"Best Accuracy {accu*100} found on randomstate {best_rstate}")
```

```
Best Accuracy 95.87897626149952 found on randomstate 88
```

```
x_train, x_test, y_train, y_test = train_test_split(X,y,test_size = .25, random_state = best_rstate,stratify=y)
```

After selecting the best random state parameter, I have spitted the data into test and train with test size as 25 %. Again, I have imported the required libraries to import my ML algorithms.

Selecting the Best model for Training

```
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import RandomizedSearchCV, cross_val_score, cross_validate, cross_val_predict
from sklearn.linear_model import SGDClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier, GradientBoostingClassifier
from sklearn.tree import DecisionTreeClassifier, ExtraTreeClassifier
from sklearn import metrics
import xgboost as xgb

import warnings
warnings.filterwarnings('ignore')
```

```
def sort_mod(estimator, x_train, y_train, cv=5, verbose=True):

    scoring = {"accuracy": "accuracy",
               "precision": "precision_weighted",
               "recall": "recall_weighted",
               "f1": "f1_weighted"}
    scores = cross_validate(estimator, x_train, y_train, cv=cv, scoring=scoring)
    accuracy, accuracy_std = scores['test_accuracy'].mean(),scores['test_accuracy'].std()
    precision, precision_std = scores['test_precision'].mean(),scores['test_precision'].std()
    recall, recall_std = scores['test_recall'].mean(),scores['test_recall'].std()
    f1, f1_std = scores['test_f1'].mean(),scores['test_f1'].std()


    ScoRes = {"Accuracy": accuracy,"Accuracy std": accuracy_std,"Precision": precision,"Precision std": precision_std,
              "Recall": recall,"Recall std": recall_std,"f1": f1,"f1 std": f1_std,}

    if verbose:
        print(f"Accuracy: {accuracy} - (std: {accuracy_std})")
        print(f"Precision: {precision} - (std: {precision_std})")
        print(f"Recall: {recall} - (std: {recall_std})")
        print(f"f1: {f1} - (std: {f1_std})")

    return ScoRes
```

Run and evaluate selected models

As we can see above, I have imported 8 classification algorithms and I am going to shortlist the best amongst these in basis of accuracy precision recall and F1 scores.

```
models = [LogisticRegression(), RandomForestClassifier(random_state=42),
          DecisionTreeClassifier(random_state=42), ExtraTreeClassifier(random_state=42),
          AdaBoostClassifier(random_state=42), GradientBoostingClassifier(random_state=42),
          xgb.XGBClassifier()]

model_names = ["LogisticRegression", "Random Forest",
               "Decision Tree", "Extra Tree", "Ada Boost",
               "Gradient Boosting", "XGBoost"]
```

```
accuracy = []
precision = []
recall = []
f1 = []

for model in range(len(models)):
    print(f"\n\nStep {model+1} of {len(models)}")
    print(f".....................................................................running {model_names[model]}")

    clf_scores = sort_mod(models[model], x_train, y_train)

    accuracy.append(clf_scores["Accuracy"])
    precision.append(clf_scores["Precision"])
    recall.append(clf_scores["Recall"])
    f1.append(clf_scores["f1"])
```

So, like above I have run all the algorithms with the data.

Key-Metrics for success in solving problem under consideration.

I have taken key metrics as Accuracy, Precision, Recall and F1 scores to analysis the best model.

	Model	accuracy	precision	recall	f1
2	Random Forest	0.955865	0.953608	0.955865	0.953448
0	LogisticRegression	0.954603	0.953689	0.954603	0.950103
7	XGBoost	0.952957	0.951311	0.952957	0.948529
3	Decision Tree	0.940340	0.939877	0.940340	0.940099
5	Ada Boost	0.945353	0.942386	0.945353	0.939511
6	Gradient Boosting	0.940022	0.940399	0.940022	0.929798
4	Extra Tree	0.920729	0.921951	0.920729	0.921303
1	Naive Bayes Gaussian	NaN	NaN	NaN	NaN

As we can see above Random Forest tops the chart, I have selected Random Forest model as my final model and I have saved the same for the further usage. Further I have imported the test data and have done prediction on the same.

```
X_test = tf_vec.fit_transform(df_test['comment_text'])
```

```
Malignant_classifier= joblib.load('Malignant_classifier.obj')
predi= Malignant_classifier.predict(X_test)
```

```
Predicted=pd.DataFrame({"Malignant_classifier":predi})
Predicted.head()
```

	Malignant_classifier
0	0
1	0
2	0
3	0
4	0

CONCLUSION

Key Findings and Conclusions of the Study

The finding of the study is that only few users over online use unparliamentary language. And most of these sentences have more stop words, and are being long. As discussed before few motivated disrespectful crowds uses these foul languages in the online forum to bully the people around and to stop them from doing the things that they are suppose to do. Our Study helps the online forms and social media to induce a ban to profanity or usage of profanity over these forms.

Learning Outcomes of the Study in respect of Data Science

The use of social media is the most common trend among the activities of today's people. Social networking sites offer today's teenagers a platform for communication and entertainment. They use social media to collect more information from their friends and followers. The vastness of social media sites ensures that not all of them provide a decent environment for children. In such cases, the impact of the negative influences of social media on teenage users increases with an increase in the use of **offensive language** in social conversations. This increase could lead to **frustration**, **depression** and a large change in their behaviour. Hence, I propose a novel approach to classify bad language usage in text conversations. I have considered the English medium for textual conversation. I have developed our system based on a foul language classification approach; it is based on an improved version of a Random Forest Classification Algorithm that detects offensive language usage in a conversation. As per our evaluation, we found that lesser number of users conversation is not decent all the time. We trained 159571 observations for eight context

categories using a Random Forest algorithm for context detection. Then, the system classifies the use of foul language in one of the trained contexts in the text conversation. In our testbed, we observed 10% of participants used foul language during their text conversation. Hence, our proposed approach can identify the impact of foul language in text conversations using a classification technique and emotion detection to identify the foul language usage

Limitations of this work and Scope for Future Work

The limitation of the study is that we have a imbalanced data so our model learnt more about the non-abusive sentence more than the abusive sentence. Which makes our model act like a overfit model when tested with live data. And also, model tend to not identify a foul or a sarcastically foul language.