MALIGNANT COMMENTS CLASSIFICATION PROJECT

Submitted by:

SATYAM TRIPATHI

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 Modelling of the Problem

Importing Libraries for preprocessing and feature engineering

```
import pandas as pd
import numpy as np
!pip install langdetect
import langdetect
```

```
from textblob import TextBlob
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
```

```
data=pd.read_csv('/content/drive/MyDrive/datatraine/train.csv')
```

```
data
```

	id	comment_text	malignant	highly_malignant	rude	threat	abuse	loathe
0	0000997932d777bf	Explanation\nWhy the edits made under my usern...	0	0	0	0	0	0
1	000103f0d9cfb60f	D'aww! He matches this background colour I'm s...	0	0	0	0	0	0
2	000113f07ec002fd	Hey man, I'm really not trying to edit war. It...	0	0	0	0	0	0
3	0001b41b1c6bb37e	"\nMore\nI can't make any real suggestions on ...	0	0	0	0	0	0
4	0001d958c54c6e35	You, sir, are my hero. Any chance you remember...	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...
159566	ffe987279560d7ff	":::::And for the second time of asking, when ...	0	0	0	0	0	0
159567	ffea4adeee384e90	You should be ashamed of yourself \n\nThat is ...	0	0	0	0	0	0
159568	ffee36eab5c267c9	Spitzer \n\nUmm, theres no actual article for ...	0	0	0	0	0	0
159569	fff125370e4aaaf3	And it looks like it was actually you who put ...	0	0	0	0	0	0
159570	fff46fc426af1f9a	"\nAnd ... I really don't think you understand...	0	0	0	0	0	0

159571 rows × 8 columns

Loading dataset and checking no. of rows and columns

As we can see there are comments that contains many things like emoji and unwanted texts like stopwords and all so we cant feed this things to our model because if I am going to feed all this it will increase the model complexity and overall these words are not important for my mode.

Ex-if I will give you a example like “You are really a awesome person”

Now my model will focus on ‘awesome’ word here so basically awesome and person word are important for this sentence.

Cleaning Text

```
import nltk
from nltk.stem import WordNetLemmatizer
from nltk.corpus import stopwords
nltk.download('stopwords')
nltk.download('wordnet')
import re
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Unzipping corpora/wordnet.zip.
```

```
wnl=WordNetLemmatizer()
corpus=[]
```

```
for i in range(len(data)):
  review=re.sub('[^a-zA-Z]',' ',data['comment_text'][i])
  review=review.lower()
  review=review.split()
  review=[wnl.lemmatize(word) for word in review if not word in stopwords.words('english')]
  review=' '.join(review)
  corpus.append(review)
```

Here i am removing all the extra text that is not useful for my model

```
data['clean_comment_text']=corpus
```

This code will help me to clean my all text that are present in this dataset.

Feature Engineering-

Here I am doing feature engineering so I can understand my data more

Like here I am adding a column that is language which will show me which comment is belong to which language and it will also help me to understand what is the percentage of comments text that belong to which language.

Feature Engineering

```
data['Language']=data['clean_comment_text'].apply(lambda x: langdetect.detect(x) if x.strip() !="" else "")
data
```

	comment_text	malignant	highly_malignant	rude	threat	abuse	loathe	clean_comment_text	Language
0	Explanation\nWhy the edits made under my usern...	0	0	0	0	0	0	explanation edits made username hardcore metal...	en
1	D'aww! He matches this background colour I'm s...	0	0	0	0	0	0	aww match background colour seemingly stuck th...	en
2	Hey man, I'm really not trying to edit war. It...	0	0	0	0	0	0	hey man really trying edit war guy constantly ...	en
3	"\nMore\nI can't make any real suggestions on ...	0	0	0	0	0	0	make real suggestion improvement wondered sect...	en
4	You, sir, are my hero. Any chance you remember...	0	0	0	0	0	0	sir hero chance remember page	en
...	...	...	...	...	...	...	...	...	...
159566	":::::And for the second time of asking, when ...	0	0	0	0	0	0	second time asking view completely contradicts...	en
159567	You should be ashamed of yourself \n\nThat is ...	0	0	0	0	0	0	ashamed horrible thing put talk page	en
159568	Spitzer \n\nUmm, theres no actual article for ...	0	0	0	0	0	0	spitzer umm there actual article prostitution ...	en
159569	And it looks like it was actually you who put ...	0	0	0	0	0	0	look like actually put speedy first version de...	en
159570	"\nAnd ... I really don't think you understand...	0	0	0	0	0	0	really think understand came idea bad right aw...	en

Here also I am adding more columns that you can see in this snapshot.

```
data['word_count']=data['clean_comment_text'].apply(lambda x:len(str(x).split(" ")))
data['char_count']=data['clean_comment_text'].apply(lambda x: sum(len(word) for word in str(x).split(" ")))
data['sentence_count']=data['clean_comment_text'].apply(lambda x: len(str(x).split(".")))
data['average_word_length']=data['char_count']/data['word_count']
data['avg_sentence_length']=data['word_count']/data['sentence_count']
```

More Feature Engineering-

Here I am adding sentiment polarity as we know in this dataset we have many sentiments like malignant, highly malignant, rude, abuse and all**.** So basically I am checking here how many are positive and negative and neutral.

```
def find_pol(review):
  return TextBlob(review).sentiment.polarity
```

inding the positive or negative sentiment with the help of TEXTBLOB

```
data['Sentiment']=data['clean_comment_text'].apply(find_pol)
```

```
def labels(review):
  if review>0:
    return 'positive'

  elif review<0:
    return 'negative'
  elif review==0:
    return 'neutral'
```

```
data['Sentiment_Label']=data['Sentiment'].apply(labels)
data.head()
```

loathe	clean_comment_text	Language	word_count	char_count	sentence_count	average_word_length	avg_sentence_length	Sentiment	Sentiment_Label
0	explanation edits made username hardcore metal...	en	23	134	1	5.826087	23.0	0.136364	positive
0	aww match background colour seemingly stuck th...	en	10	58	1	5.800000	10.0	0.250000	positive
0	hey man really trying edit war guy constantly ...	en	21	121	1	5.761905	21.0	0.150000	positive
0	make real suggestion improvement wondered sect...	en	52	313	1	6.019231	52.0	0.257143	positive
0	sir hero chance remember page	en	5	25	1	5.000000	5.0	0.000000	neutral

As we can see new column is been added in this dataset that is sentiment labels.

Checking Count of Each Comments-

As we can see there are comments labels which are imbalanced if I am going to feed this to my model my model will not able to understand about other classes which have less count so I need to balance all the things but first I am doing EDA.

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

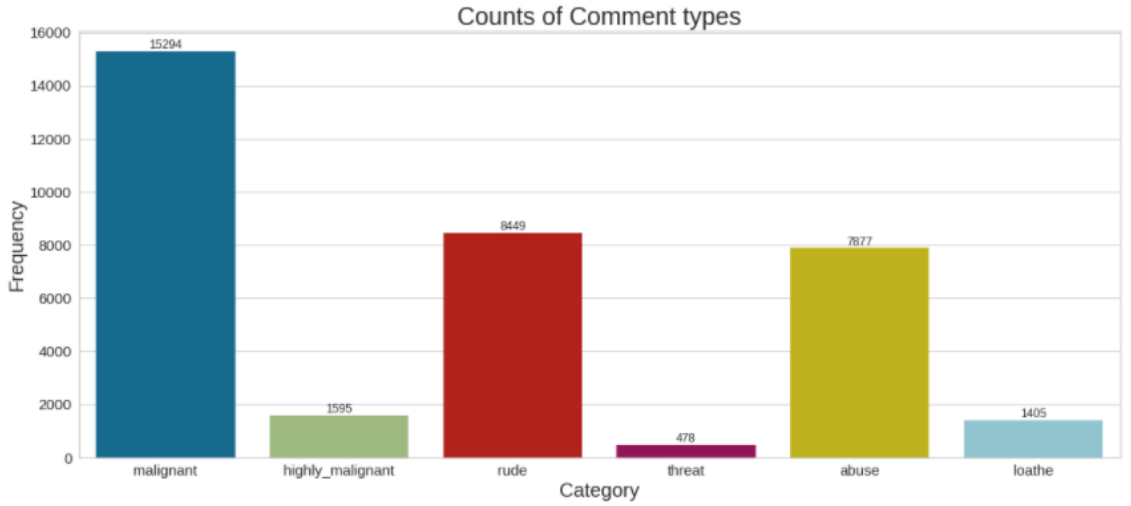
```
counts=df.sum()
counts
```

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

EDA-

EDA

```
plt.figure(figsize=(19,8))
ax=sns.barplot(counts.index,counts.values)
plt.title("Counts of Comment types",fontsize=25)
plt.ylabel("Frequency",fontsize=20)
plt.yticks(fontsize=15)
plt.xlabel("Category",fontsize=20)
plt.xticks(fontsize=15)
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()
```

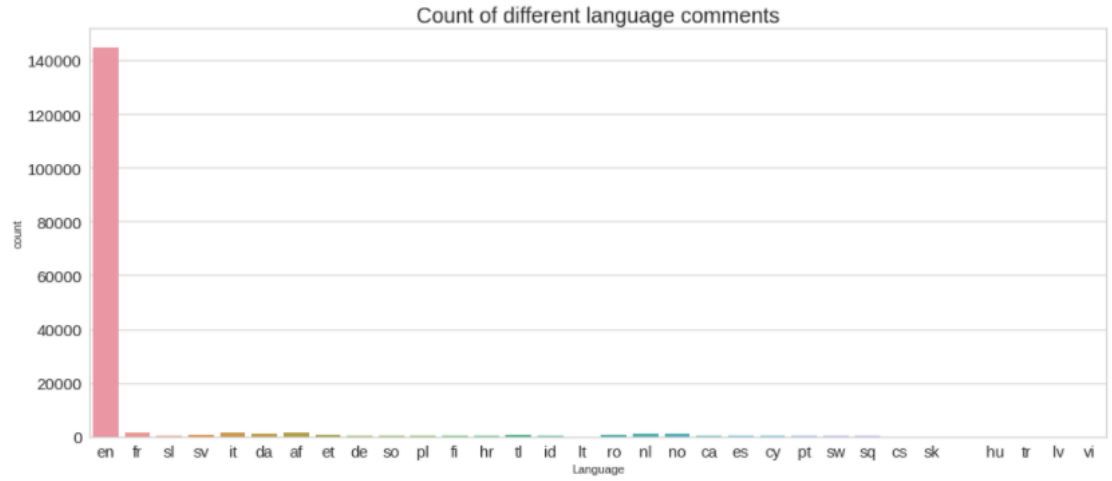


As we can see this Countplot shows us that malignant has the high count and threat and loathe has less count means mostly of the people who comment that belong to malignant class.

CountPlot Of Language-

```
plt.figure(figsize=(17,7))
sns.countplot(data['Language'])
plt.title("Count of different language comments",fontsize=20)
plt.xticks(fontsize=15)
plt.yticks(fontsize=15)
```

```
(array([     0.,  20000.,  40000.,  60000.,  80000., 100000., 120000.,
        140000., 160000.]), <a list of 9 Text major ticklabel objects>)
```

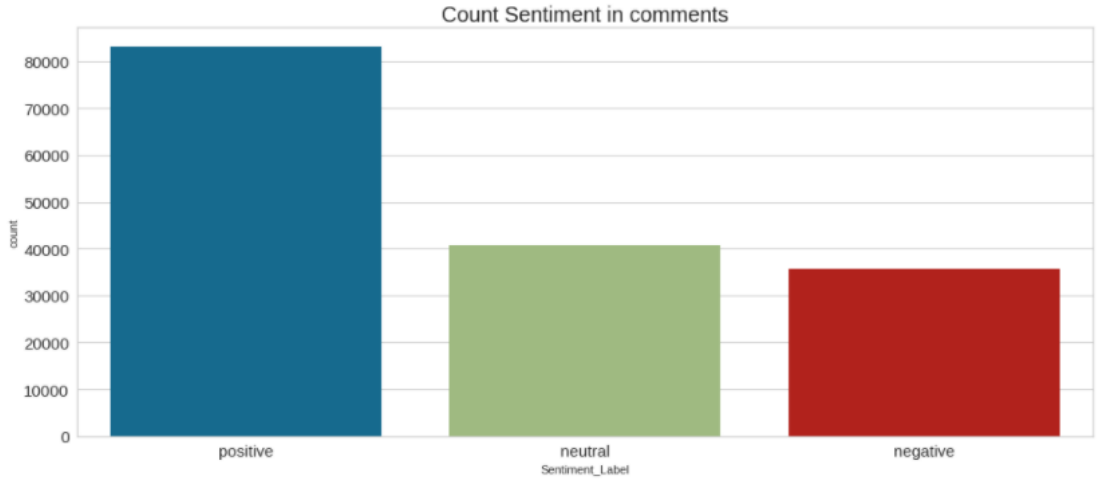


As we can see there are many comments that belong to only English and very few comment that belongs to other languages.

Countplot of Sentiments-

```
plt.figure(figsize=(17,7))
sns.countplot(data['Sentiment_Label'])
plt.title("Count Sentiment in comments",fontsize=20)
plt.xticks(fontsize=15)
plt.yticks(fontsize=15)
```

```
(array([    0., 10000., 20000., 30000., 40000., 50000., 60000., 70000.,
        80000., 90000.]), <a list of 10 Text major ticklabel objects>)
```

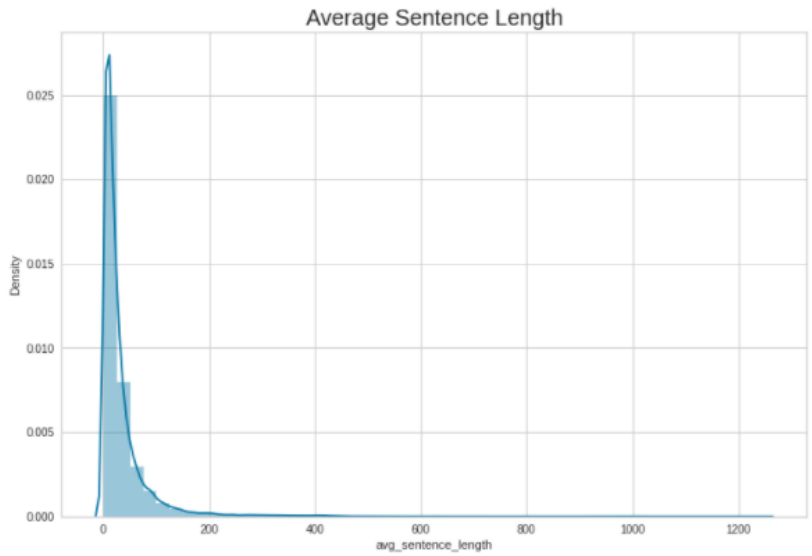


As we can see there are mostly comments that are positive and then mostly comments are neural after then we have negative comments and in negative we can many labels like malignant, highly malignant, rude, abuse and all.

Average Sentence Length-

```
: plt.figure(figsize=(12,8))
  sns.distplot(data['avg_sentence_length'])
  plt.title("Average Sentence Length",fontsize=20)
```

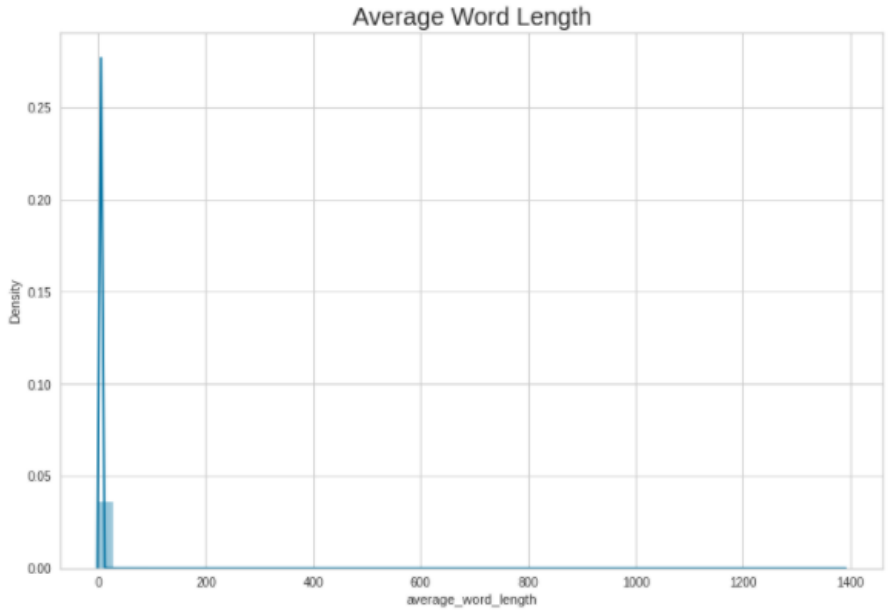
```
: Text(0.5, 1.0, 'Average Sentence Length')
```



The average length of comment lies between 0-50 and there are also many peoples who use to do long comments like 1200 long also.

Average Word Length-

```
: plt.figure(figsize=(12,8))
  sns.distplot(data['average_word_length'])
  plt.title("Average Word Length",fontsize=20)

: Text(0.5, 1.0, 'Average Word Length')
```



The average word length of most of the comments is between 0-20 with the above graph i can see which comment type is having more count.

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.

words from malignant comment

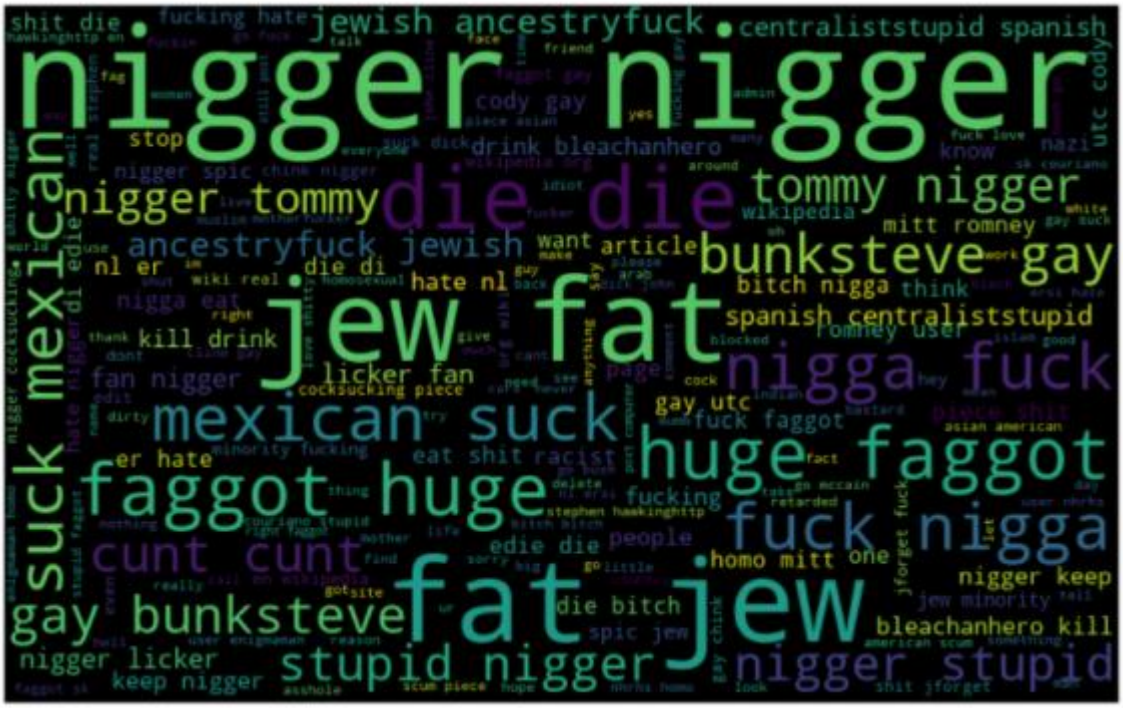
```
from wordcloud import WordCloud
all_words = " ".join([sentence for sentence in data['clean_comment_text'][data['malignant']==1]])

wordcloud = WordCloud(width=800, height=500, random_state=42, max_font_size=100).generate(all_words)

# plot the graph
plt.figure(figsize=(15,8))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.show()
```

Words From Highly Malignant Comments

Word from Threat Comments

Words from Rude comments

Word from loathe comments



These are the comments that belongs to different type so which the help of wordcloud we can see if there is abuse comment which type of words it contains and similar to other comments as well.

Data Augmentation-

As we have seen above we have a total imbalanced dataset so I have to balance it so my model can understand each and every comment label so for the I will be using NLP Data Augmentation Techniques like Transformer Models.

```
: import torch
  from transformers import PegasusForConditionalGeneration, PegasusTokenizer

  model_name = 'tuner007/pegasus_paraphrase'
  torch_device = 'cuda' if torch.cuda.is_available() else 'cpu'
  tokenizer = PegasusTokenizer.from_pretrained(model_name)
  model = PegasusForConditionalGeneration.from_pretrained(model_name).to(torch_device)
```

```
def get_response(input_text,num_return_sequences):
  batch = tokenizer.prepare_seq2seq_batch([input_text],truncation=True,padding='longest',max_length=60, return_tensors="pt").to(torch_dev
ice)
  translated = model.generate(**batch,max_length=60,num_beams=10, num_return_sequences=num_return_sequences, temperature=1.5)
  tgt_text = tokenizer.batch_decode(translated, skip_special_tokens=True)
  return tgt_text
```

```
df=threat_data['clean_comment_text'][0:20]
```

```
for i in df:
  print(get_response(i,5))
```

```
def highly_malignant1(df):
    new_text=[]

    ##selecting the minority class samples
    df_n=df[df.highly_malignant==1].reset_index(drop=True)

    ## data augmentation loop
    for i in tqdm_notebook(range(0,len(df_n))):

            text = df_n.iloc[i]['clean_comment_text']
            augmented_text = get_response(text,9)
            for i in augmented_text:
              new_text.append(i)


    ## dataframe
    new=pd.DataFrame({'clean_comment_text':new_text,'highly_malignant':1})
    return new
```

```
highly_malignant=highly_malignant1(data)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:8: TqdmDeprecationWarning: This function will be removed in tqdm==5.0.0
Please use `tqdm.notebook.tqdm` instead of `tqdm.tqdm_notebook`
```

```
  0%|          | 0/1595 [00:00<?, ?it/s]
```

Which the Help of this code and transformers library and model I will be able to create many other comments that are similar to the original comments.

```
highly_malignant.drop_duplicates()
```

	clean_comment_text	highly_malignant
0	The cocksucker is around work.	1
1	cocksucker is around work.	1
2	cocksucker is at work.	1
3	The cocksucker is at work.	1
4	cocksucker is around work	1
...	...	...
14350	Tell anyone that rape jew family choke bagel, ...	1
14351	Tell anyone that rape jew family choke bagel, ...	1
14352	Tell anyone that rape jew family choke bagel, ...	1
14353	Tell anyone that rape jew family choke bagel, ...	1
14354	Tell anyone that rape jew family choke bagel a...	1

As we can see in this upper snapshot of the comments this is how transformer model is able to create similar kind of text so after this I can add more flexibility to the model.

Same thing I have did with other labels as well like some labels was having only 500 count like threat so I have generated 15 texts from one sentence so I will be equal to the higher count of label like malignant comment.

So finally at the end after augmentation I was able to create similar kind of text now all my labels have approx. 15000 counts.

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. NLTK
9. Transformers
10. Google colab

Model/s Development and Evaluation

Training Models-

```
from sklearn.model_selection import train_test_split,cross_val_score,StratifiedKFold,GridSearchCV
from sklearn.metrics import accuracy_score,classification_report,f1_score,roc_auc_score,log_loss
```

```
list_classes = ['malignant','highly_malignant','rude','threat','abuse','loathe']
train_y = data[list_classes].values
X_train,X_test,y_train,y_test=train_test_split(train_tfidf,train_y,test_size=0.3,random_state=42)
```

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from xgboost import XGBClassifier
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.neighbors import KNeighborsClassifier
```

```
models={"RandomForestClassifier":RandomForestClassifier(),
        "DecisionTreeClassifier":DecisionTreeClassifier(),
        "ExtraTreesClassifier":ExtraTreesClassifier(),
        "KNeighborsClassifier":KNeighborsClassifier()}
```

```
for name,model in models.items():
  print("**************************************************",name,"**************************************************")
  print('\n')
  model.fit(X_train,y_train)
  print(model)
  y_pred=model.predict(X_test)
  print('\n')
  acc=accuracy_score(y_test,y_pred)
  print("Accuracy Score",acc)
  print('\n')
  F1_SCORE=f1_score(y_test,y_pred,average='micro')
  print("F1 SCORE",F1_SCORE)
  report=classification_report(y_test,y_pred)
  print('\n')
  print("CLASSIFICATION REPORT")
  print("\n")
  print(report)
  roc=roc_auc_score(y_test,y_pred)
  print("ROC AUC SCORE",roc)
```

```
************************************************** RandomForestClassifier **************************************************
```

Here I have used some tree base model for better accuracy.

And from all of the above model ExtraTreesClassifier was giving me good accuracy so I have picked it as a final model.

Final Model Performance Metrics

Final Model Performance Metrics

```
print("MSE",accuracy_score(y_test,pre))
print("F1 Score",f1_score(y_test,pre,average='micro'))
print("Classification Report")
print(classification_report(y_test,pre))
print("Log Loss",log_loss(y_test,pre))
```

```
MSE 0.8186698088504845
F1 Score 0.8093076642145355
Classification Report
              precision    recall  f1-score   support

           0       0.92      0.78      0.84      9713
           1       0.95      0.84      0.89      1487
           2       0.81      0.73      0.77      9687
           3       0.91      0.87      0.89      4725
           4       0.80      0.71      0.75      9168
           5       0.87      0.79      0.83      5680

   micro avg       0.86      0.77      0.81     40460
   macro avg       0.87      0.79      0.83     40460
weighted avg       0.86      0.77      0.81     40460
 samples avg       0.28      0.28      0.28     40460

Log Loss 3.652755850341734
```

Note- As I am having a very big data my system was not able to handle it so I was not able to do hyperparameter tuning.

Test Data-

Loading the test data and cleaning it for prediction.

Laoding Test Data

```
test=pd.read_csv('/content/drive/MyDrive/datatraine/test.csv')
```

Cleaning Test Data

```
wnl=WordNetLemmatizer()
corpus=[]

for i in range(len(test)):
  review=re.sub('[^a-zA-Z]',' ',test['comment_text'][i])
  review=review.lower()
  review=review.split()
  review=[wnl.lemmatize(word) for word in review if not word in stopwords.words('english')]
  review=' '.join(review)
  corpus.append(review)
```

```
test['clean_comment_text']=corpus
```

Feature Transformation

```
tfidf_vec = TfidfVectorizer(ngram_range=(1,2),
                            min_df=2,
                            max_features=15000)
tfidf_vec.fit(test['clean_comment_text'])

# trasform train and test
test_tfidf = tfidf_vec.transform(test['clean_comment_text'])
```

```
predictions=Extra.predict(test_tfidf)
```

```
pre_data=pd.DataFrame(predictions,columns=list_classes)
```

```
test_data_comments=pd.concat([pre_data,test['clean_comment_text']],axis=1)
```

Now here I have predicted the text and created a new dataframe of predicted labels so I can understand the model predictions.

Predicted Results For Malignant comments and others-

The result of prediction of malignant comment

```
test_data_comments[test_data_comments['malignant']==1]
```

	malignant	highly_malignant	rude	threat	abuse	loathe	clean_comment_text
14	1.0	0.0	0.0	0.0	0.0	0.0	adding new product list make sure relevant add...
41	1.0	0.0	0.0	0.0	0.0	0.0	convinced blind documented possible call legal...
75	1.0	0.0	0.0	0.0	1.0	0.0	ridiculous unless good non disingenuous respon...
100	1.0	0.0	1.0	0.0	0.0	0.0	could precis material instead downloading whol...
104	1.0	0.0	0.0	0.0	0.0	0.0	catdiffuse think best idea would check whatlin...
...	...	...	...	...	...	...	...
153038	1.0	0.0	0.0	0.0	0.0	0.0	rate article definitely need work capitalizati...
153061	1.0	0.0	0.0	0.0	0.0	0.0	sarah anyone edit page anyone nominate deletio...
153106	1.0	0.0	1.0	0.0	0.0	0.0	youshit dick cock fuckshit dick cock fuck shit...
153116	1.0	0.0	1.0	0.0	0.0	0.0	utc fyi currently edit conflict user bakasuprm...
153159	1.0	0.0	1.0	0.0	1.0	0.0	totally agree stuff nothing long crap

7466 rows × 7 columns

Highly_malignant comment

]:
```
test_data_comments[test_data_comments['highly_malignant']==1]
```

]:

	malignant	highly_malignant	rude	threat	abuse	loathe	clean_comment_text
62046	1.0	1.0	1.0	0.0	1.0	0.0	p user also editing people talk page
67203	1.0	1.0	1.0	0.0	1.0	0.0	changing people talk page comment

Abuse Comment

]:
```
test_data_comments[test_data_comments['abuse']==1]
```

]:

	malignant	highly_malignant	rude	threat	abuse	loathe	clean_comment_text
75	1.0	0.0	0.0	0.0	1.0	0.0	ridiculous unless good non disingenuous respon...
106	0.0	0.0	0.0	0.0	1.0	0.0	avg plenty greek love king stop acting like gr...
288	1.0	0.0	1.0	0.0	1.0	0.0	recommendation point keep stuff main space
308	1.0	0.0	1.0	0.0	1.0	0.0	february utc certainly would ask include somet...
382	1.0	0.0	1.0	0.0	1.0	0.0	related page fyi exists draft draft trumpler c...
...	...	...	...	...	...	...	...
152654	1.0	0.0	0.0	0.0	1.0	0.0	page movement hello jk getting impression thar...
152792	1.0	0.0	1.0	0.0	1.0	0.0	tripped keyboard oops tripped keyboard wrote t...
153041	0.0	0.0	1.0	0.0	1.0	0.0	closed captioning read nigger
153090	0.0	0.0	1.0	0.0	1.0	0.0	dear sir fucking cunt rahm emanuel satannnnnnn...
153159	1.0	0.0	1.0	0.0	1.0	0.0	totally agree stuff nothing long crap

2745 rows × 7 columns

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 190000 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.