MALIGNANT COMMENTS CLASSIFER

Submitted by:

BHAVNA PIPLANI

ACKNOWLEDGMENT

This includes mentioning of all the references, research papers, data sources, professionals and other resources that helped and guided me in completion of the project.

INTRODUCTION

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 behavior.

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 inoffensive, 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.

Description of the domain related concepts that will be useful for better understanding of the project.

Data Cleaning, Data Visualisation using different plotting methods like barplot, countplot, wordcloud,distplot and, Data pre-processing using TfIdf vectorizer and pipelining for Model Training

B. Analytical Problem Framing

a) Mathematical/ Analytical Modelling of the Problem

Statistical modelling is the process of applying statistical analysis to a dataset. A statistical model is a mathematical representation (or mathematical model) of observed data.

When data analysts apply various statistical models to the data they are investigating, they are able to understand and interpret the information more strategically. Rather than sifting through the raw data, this practice allows them to identify relationships between variables, make predictions about future sets of data, and visualize that data so that non-analysts and stakeholders can consume and leverage it.

most common techniques will fall into the following two groups:

Supervised learning, including regression and classification models.

Unsupervised learning, including clustering algorithms and association rules.

b) Data Sources and their formats

Data Set Description

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.

C. Explanatory Data Analysis

```
1 import pandas as pd
2 import numpy as np
3 import seaborn as sns
4 import matplotlib.pyplot as plt
5 import warnings
6 warnings.simplefilter("ignore")
```

Importing the dataset from the source file provided.

```
1 train_data = pd.read_csv("traincomment.csv",sep='\t')
2 train_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	0.0	0.0	0.0
1	000103f0d9cfb60f	D'aww! He matches this background colour I'm s...	0.0	0.0	0.0	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	0.0	0.0	0.0
3	0001b41b1c6bb37e	"\nMore\nI can't make any real suggestions on ...	0.0	0.0	0.0	0.0	0.0	0.0
4	0001d958c54c6e35	You, sir, are my hero. Any chance you remember...	0.0	0.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...
159616	ffe987279560d7ff	":::::And for the second time of asking, when ...	0.0	0.0	0.0	0.0	0.0	0.0

After importing the dataset, checking the datatypes of each column.

```
1 train_data.dtypes   # checking the datatypes of cols
```

```
id                     object
comment_text           object
malignant             float64
highly_malignant      float64
rude                  float64
threat                float64
abuse                 float64
loathe                float64
dtype: object
```

Train dataset has 159621 rows and 8 columns

```
1 train_data.shape  # checking the rows and cols count
```

```
(159621, 8)
```

After checking the no. of rows and columns count, we will check the null values if any, column count, datatypes of columns.

```
1 train_data.isnull().sum() # null  values column wise counts
```

```
id                     0
comment_text          74
malignant             93
highly_malignant      93
rude                  93
threat                93
abuse                 93
loathe               112
dtype: int64
```

The dataset is not clean. Data is missing in many of the features. There are missing or null values in the dataset. We will drop the null rows.

```
# dropping rows having null values
train_data=train_data.dropna()
train_data
```

After checking for null values , we will see the values count for each class .

```
comment_count=train_data.iloc[:,2:].sum()
comment_count
```

```
malignant           15289.0
highly_malignant     1594.0
rude                 8444.0
threat                478.0
abuse                7875.0
loathe               1405.0
dtype: float64
```

Above image shows that data is imbalanced. As the comments for class malignant is very high as compared to other class. So, we will use class balancing technique SMOTE over sampling to balance the data and to avoid the situation of overfitting.

D. Data Preprocessing

Comments contains lots of spacing, special characters, numerical text etc. We will use different ways to clean the text used in one function shown below:

```
# function to do all preprocessing on comments
def clean_comment(comments_text):
    comments_text=re.sub(r'http\$+','',comments_text) # removing the url
    comments_text=re.sub('[^a-zA-Z]',' ',comments_text) #removing Numbers and punctuation
    comments_text=str(comments_text).lower().replace('\\','').replace('_',' ') #converting
    comments_text=word_tokenize(comments_text) #tokenization
    comments_text=[item for item in comments_text if item not in stop_words] # removing sto
    comments_text=[lemma.lemmatize(word=w,pos='v') for w in comments_text] #lemmatization
    comments_text=[i for i in comments_text if len(i)>=2] # removing the words having lengt
    return comments_text
```

Applying the above function on train_data

```
train_data['comment_text']=train_data["comment_text"].apply(lambda x:clean_comment(x)) # pr
```

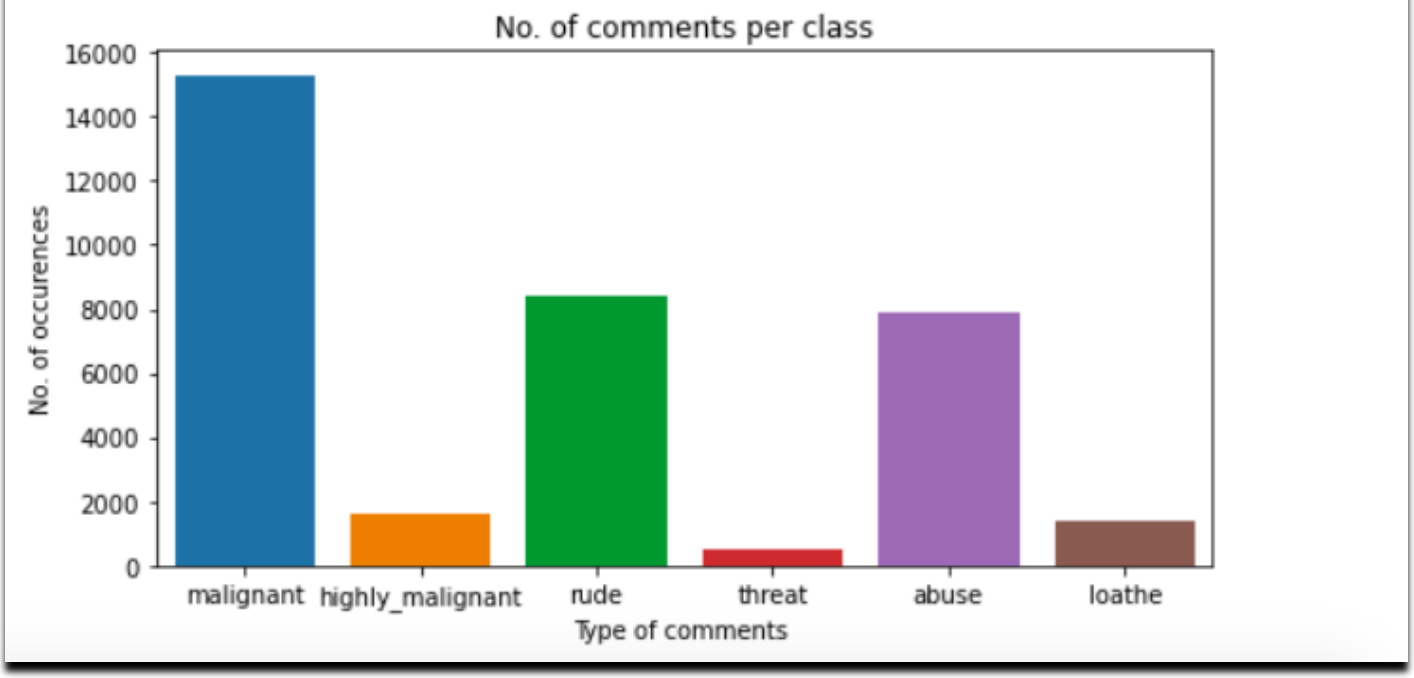
```
train_data['comment_text']=[" ".join(comments_text) for comments_text in train_data['commen
train_data
```

	id	comment_text	malignant	highly_malignant	rude	threat	abuse	loathe
0	0000997932d777bf	explanation edit make username hardcore metall...	0.0	0.0	0.0	0.0	0.0	0.0
1	000103f0d9cfb60f	aww match background colour seemingly stick th...	0.0	0.0	0.0	0.0	0.0	0.0
2	000113f07ec002fd	hey man really try edit war guy constantly rem...	0.0	0.0	0.0	0.0	0.0	0.0
3	0001b41b1c6bb37e	make real suggestions improvement wonder secti...	0.0	0.0	0.0	0.0	0.0	0.0
4	0001d958c54c6e35	sir hero chance remember page	0.0	0.0	0.0	0.0	0.0	0.0

Till here, same EDA was done on test data .

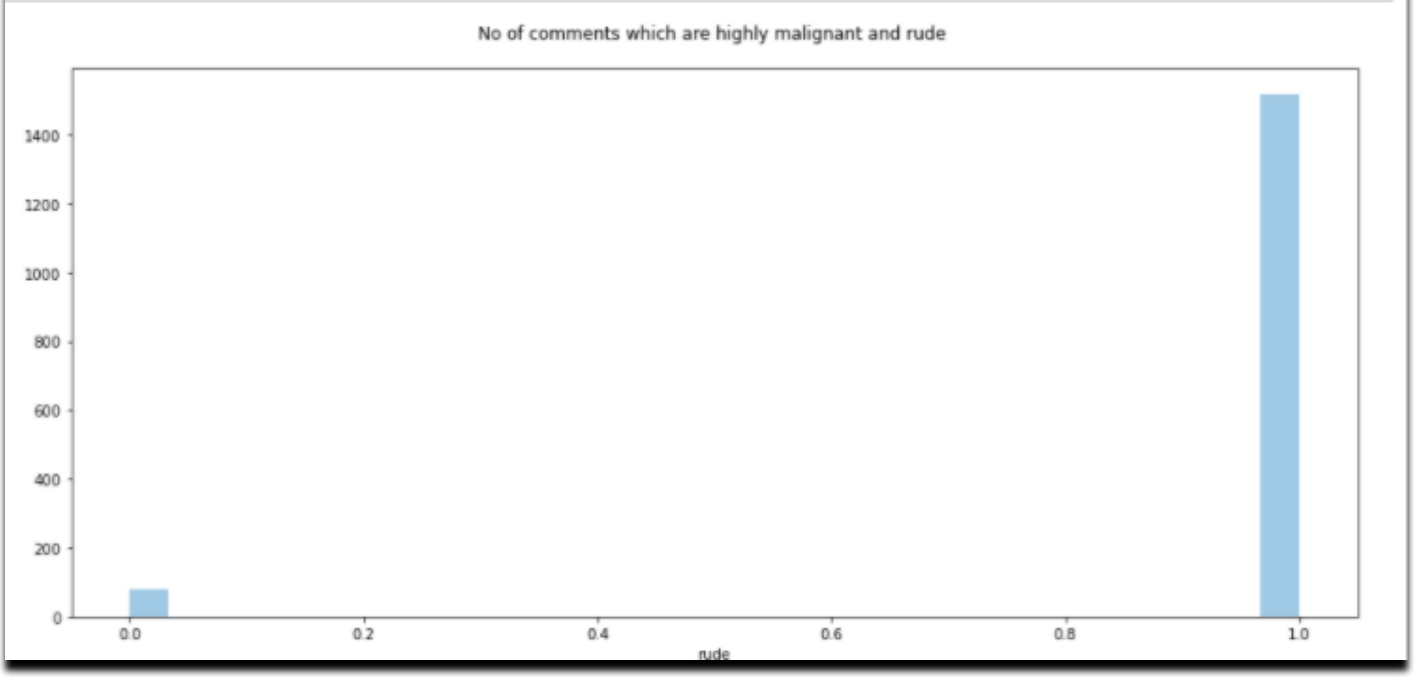
Data Visualization

```
plt.figure(figsize=(8,4))
ax=sns.barplot(comment_count.index,comment_count.values)

plt.title("No. of comments per class")
plt.ylabel("No. of occurences")
plt.xlabel("Type of comments ")

plt.show()
```



Above countplot shows that no of comments per class. As it shows a very high range in malignant class and very low comments in threat class.

```
plt.figure(figsize=(16,7))
sns.distplot(train_data[train_data['highly_malignant']==1]['rude'])
plt.title('No of comments which are highly malignant and rude \n')
plt.show()
```



Above distplot shows around 90 comments are highly malignant and rude too.

```
def wordcloud(comment_count,label):
    # finding classes with malignant comments only
    subset=comment_count[comment_count[label]==1]
    text=subset.comment_text.values
    wc=WordCloud(background_color='black',max_words=4000)

    wc.generate(" ".join(text))

    plt.figure(figsize=(20,20))
    plt.subplot(221)
    plt.axis("off")
    plt.imshow(wc.recolor(colormap='gist_earth',random_state=244),
```

```
1 wordcloud(malig_data,"malignant")
```

Above wordcloud shows the words frequently used in malignant class.

Libraries and packages used for model training are listed below

```
1 #importing the model training libraries
2 from sklearn.model_selection import train_test_split
3 from sklearn.naive_bayes import MultinomialNB,BernoulliNB
4 from sklearn.linear_model import LogisticRegression
5 from sklearn.svm import LinearSVC
6 from sklearn.multiclass import OneVsRestClassifier
7 from sklearn.pipeline import Pipeline
```

```
1 from sklearn.metrics import f1_score,confusion_matrix,precision_score,recall_score,classifi
2 import warnings
3 warnings.filterwarnings('ignore')
```

E. Model Development and Evaluation

Identification of possible problem-solving approaches (methods)most common techniques will fall into the following two groups:

Supervised learning, including regression and classification models.

Unsupervised learning, including clustering algorithms and association rules.

For this dataset, I will be using classification model because the output variable is nominal.

Testing of Identified Approaches (Algorithms)

Here, In this project I will be using **LogisticRegression(), BernoulliNB(), MultinomialNB(), LinearSVC()** algorithms

Steps to be followed in loop for all the classes

1. (**TDidF Vectorizer**) Encoding text into vectors for further model training

2. **Pipeline** the models with TDidF Vectorizer and OneVsrestClassifier for solving multiclass classification model

3. **Model Training and metrics representation**

```
train, test = train_test_split(train_data, random_state=42, test_size=0.33, shuffle=True)
X_train = train.comment_text
X_test = test.comment_text
print(X_train.shape)
print(X_test.shape)
```

```
(106871,)
(52638,)
```

After train test split , we will train the model

```
from sklearn.feature_extraction.text import TfidfVectorizer
# making 3 datasets to display f1_score,precion score ,recall score
f1_score_data=pd.DataFrame(index=["Log Regression","BernoulliNB","MultinomialNB","LinearSVC
precision_data=pd.DataFrame(index=["Log Regression","BernoulliNB","MultinomialNB","LinearSV
recall_data=pd.DataFrame(index=["Log Regression","BernoulliNB","MultinomialNB","LinearSVC"]

model=[LogisticRegression(),BernoulliNB(),MultinomialNB(),LinearSVC()]


for i in classes:
        #making lists to store f1_score,precision ,recall for all 4 models
    f1_score_list=[]
    precision_list=[]
    recall_list=[]
    for m in model:
        #pipelining the model with OneVsRestClassifier for multiclass classification
        NB_pipeline = Pipeline([
                ('tfidf', TfidfVectorizer(smooth_idf=False,max_features=20000,ngram_range=(
                ('clf', OneVsRestClassifier(m)),])

        print('MultinomialNB model performance for',i,'with \n' )
    # train the model
        NB_pipeline.fit(X_train,train[i])
```

```
    # compute the testing accuracy
        y_pred=NB_pipeline.predict(X_test)
    #appending metrics in respective lists
        f1_score_list.append(f1_score(test[i],y_pred,average='micro'))
        precision_list.append(precision_score(test[i],y_pred,average='micro'))
        recall_list.append(recall_score(test[i],y_pred,average='micro'))

        f1_scores_data=f1_score_list
        precisions_data=precision_list
        recalls_data=recall_list

        print("classification report for",i," comments of model" , m, "is \n",classificatio
        print("**********************************************************************\n")

    # copying all 4 models and their metrics for 6 class in datasets
    f1_score_data[i]=f1_scores_data
    precision_data[i]=precisions_data
    recall_data[i]=recalls_data
```

Results derived from the above steps are as follows:

F1_score of all 4 models used for all the classes :

```
1  f1_score_data
```

	malignant	highly_malignant	rude	abuse	threat	loathe
Log Regression	0.957141	0.990349	0.978115	0.969813	0.997207	0.992268
BernoulliNB	0.704358	0.986474	0.760401	0.773092	0.991622	0.987082
MultinomialNB	0.945097	0.990064	0.967628	0.963619	0.997017	0.991527
LinearSVC	0.959630	0.990311	0.979938	0.971124	0.997207	0.992363

Above dataframe image shows that f1_score is highest for LinearSVC model

Precision_score of all 4 models used for all the classes :

```
1 precision_data
```

	malignant	highly_malignant	rude	abuse	threat	loathe
Log Regression	0.957141	0.990349	0.978115	0.969813	0.997207	0.992268
BernoulliNB	0.704358	0.986474	0.760401	0.773092	0.991622	0.987082
MultinomialNB	0.945097	0.990064	0.967628	0.963619	0.997017	0.991527
LinearSVC	0.959630	0.990311	0.979938	0.971124	0.997207	0.992363

Above dataframe image shows that precision_score is highest for LinearSVC model.

recall_score of all 4 models used for all the classes :

```
1 recall_data
```

	malignant	highly_malignant	rude	abuse	threat	loathe
Log Regression	0.957141	0.990349	0.978115	0.969813	0.997207	0.992268
BernoulliNB	0.704358	0.986474	0.760401	0.773092	0.991622	0.987082
MultinomialNB	0.945097	0.990064	0.967628	0.963619	0.997017	0.991527
LinearSVC	0.959630	0.990311	0.979938	0.971124	0.997207	0.992363

After this , we will cross validate our best performing model i.e LinearSVC with roc_auc scoring method and will analyse the mean and standard deviation.

parameter tuning by cross validating through roc_auc scoring

```
# cross validating LinearSVC
from sklearn.model_selection import cross_val_score

SVC_pipeline = Pipeline([
                ('tfidf', TfidfVectorizer(smooth_idf=False,max_features=20000,ngram_range=(
                ('clf', OneVsRestClassifier(LinearSVC())),
            ])
for i  in classes:
    print('LinearSVC model performance for class {}'.format(i))
    # train the model using X_dtm & y
    SVC_pipeline.fit(X_train, train[i])
    # compute the testing accuracy
    prediction = SVC_pipeline.predict(X_test)
    #cross val score using scoring as roc_auc
    score=cross_val_score(SVC_pipeline,X_test,test[i],cv=4,scoring='roc_auc')
    print("Cross Validation Score for " ,i , "is : " ,score,"\n")
    print("Mean for ",i,"is : " , score.mean())
    print("Standard Deviation for " , i , "is :" , score.std())
```

Cross validation score shows very good accuracy with 4 folds at 98% .

Saving the model- Serialization

```
# saving the prediction model

import pickle
filename="comments.pkl"
pickle.dump(SVC_pipeline,open(filename,'wb'))
```

After doing serialization, Now we will pass it in fitted model and will predict the test data of 153157 rows given.

```
# load the model
fitted_model=pickle.load(open("comments.pkl",'rb'))
```

```
fitted_model
```

```
Pipeline(steps=[('tfidf',
                 TfidfVectorizer(max_features=20000, smooth_idf=False)),
                ('clf', OneVsRestClassifier(estimator=LinearSVC()))])
```

```
# predictions over test data (testcomment.csv)

predictions=fitted_model.predict(test_data["comment_text"])
```

Lets save the fitted model in csv file.

```
#prediction = pd.DataFrame(predictions, columns=['predictions']).to_csv('prediction.csv')
ds_pred=pd.DataFrame(data=predictions,columns=["Label"]).to_csv('prediction.csv')
ds_pred
```

Lets test the fitted model.

```
vect=SVC_pipeline.predict(["bullshit"])
```

```
vect
```

```
array([1.])
```

```
vect1=SVC_pipeline.predict(["cool"])
vect1
```

```
array([0.])
```

CONCLUSION

Key Findings and Conclusions of the Study

1. Dataset was very huge requiring lot of data cleaning.
2. OneVsRestClassifier was used for multiclass classification.
3. F1 score and Recall metrics was used to see how well the positive class was predicted and is the same calculation as sensitivity.
4. Precision was used to see the fraction of examples assigned the positive class that belong to the positive class.
5. LinearSVC was the best fit model.