FLIP ROBO

MALIGNANT COMMENTS CLASSIFICATION

Submitted by:

Junaid Shaikh

ACKNOWLEDGMENT

One of the pleasant aspects of preparing a project report is the opportunity to thank those who have contributed to make this project possible.

We are extremely thankful to Mr. Tushar Saraswat, whose active interest in the project & insight helped us to formulate, redefine implement our approach to the project.

We are also thankful to our institute & Other seen unseen hands which have given us direct & indirect help in completion of this project.

- **Himanshu Sharma**

INTRODUCTION

- ## Business Problem Framing

This project is related to the scenario of 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 must 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

Conceptual Background in this dataset, i.e. We will use Statistics and machine learning algorithms as well as natural language technic to process the dataset and built the model. Since the target value is classified, we will use the classification model algorithms in it. To visualize the data, we will use matplotlib and seaborn library, which based on python language.

- ## Review of Literature

  Review of Literature is basically related to comprehensive summary of dataset as well as descriptions of input variables and output variable.

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.

	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
5	00025465d4725e87	"\n\nCongratulations from me as well, use the ...	0	0	0	0	0	0

- ## Motivation for the Problem Undertaken

  My motivation behind solving this classification problem is that it will help us to classify comments in social media.

Analytical Problem Framing

- ## Mathematical/ Analytical Modelling of the Problem

In this section we understand and describe the mathematical, statistical and analytics modelling done during this project along with the proper justification. First of all for better understanding about dataset and given attributes information we use **Dataframe.info()** command, which tell me, The total number of attributes, what is name of attributes, datatype of attributes and how many Non-null values are present in dataset.

```
df_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 159571 entries, 0 to 159570
Data columns (total 8 columns):
 #   Column            Non-Null Count   Dtype 
---  ------            --------------   ----- 
 0   id                159571 non-null  object
 1   comment_text      159571 non-null  object
 2   malignant         159571 non-null  int64 
 3   highly_malignant  159571 non-null  int64 
 4   rude              159571 non-null  int64 
 5   threat            159571 non-null  int64 
 6   abuse             159571 non-null  int64 
 7   loathe            159571 non-null  int64 
dtypes: int64(6), object(2)
memory usage: 9.7+ MB
```

After understanding the dataset values, we take the statistical descriptions of dataset using **Dataframe.describe()** command in python, which tells the following statistical descriptions:

```
df_train.describe()
```

	malignant	highly_malignant	rude	threat	abuse	loathe
count	159571.000000	159571.000000	159571.000000	159571.000000	159571.000000	159571.000000
mean	0.095844	0.009996	0.052948	0.002996	0.049364	0.008805
std	0.294379	0.099477	0.223931	0.054650	0.216627	0.093420
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
50%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
75%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

After knowing about statistical description, we move forward in the way of finding co-relation of variables, then move to data-cleaning, then move to visualized the data set and their relationship. After this we move to data pre-processing and modelling as well as testing the accuracy of model.

- ## Data Sources and their formats

Machine learning algorithms as well as Natural Language processing are almost always optimized for raw, detailed source data. Thus, the data environment must provision large quantities of raw data for discovery-oriented analytics practices such as data exploration, data mining, statistics, and machine learning.

Tabular data for machine learning is typically found is .csv files. Csv files are text-based files containing comma separated values (csv). Csv files are popular for ML as they are easy to view/debug and easy to read/write from programs (no compression/indexing).

- ## Data Pre-processing Done

  a. Read dataset and make it in proper format.
  b. Encode labels
  c. Convert all cases to lower
  d. Remove punctuations
  e. Remove Stopwords
  f. Check stats of messages
  g. Convert all texts into vectors
  h. Import classifier
  i. Train and test
  j. Check the accuracy/confusion matrix.

- ## Hardware and Software Requirements and Tools Used

  In this project dataset is too large for processing or modelling, that's why we use good hardware configuration like as above 4GB RAM, above or equal core i3 processor and need good storage HDD. In way of software, we use any operating system which support python language for coding.

Model/s Development and Evaluation

- **Identification of possible problem-solving approaches (methods)**

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

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

```
#  Convert text into vectors using TF-IDF
from sklearn.feature_extraction.text import TfidfVectorizer
tf_vec = TfidfVectorizer(max_features = 10000, stop_words='english')
features = tf_vec.fit_transform(df_train['comment_text'])
x = features
```

```
y=df_train['bad']
x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=56,test_size=.30)
```

Testing of Identified Approaches (Algorithms)

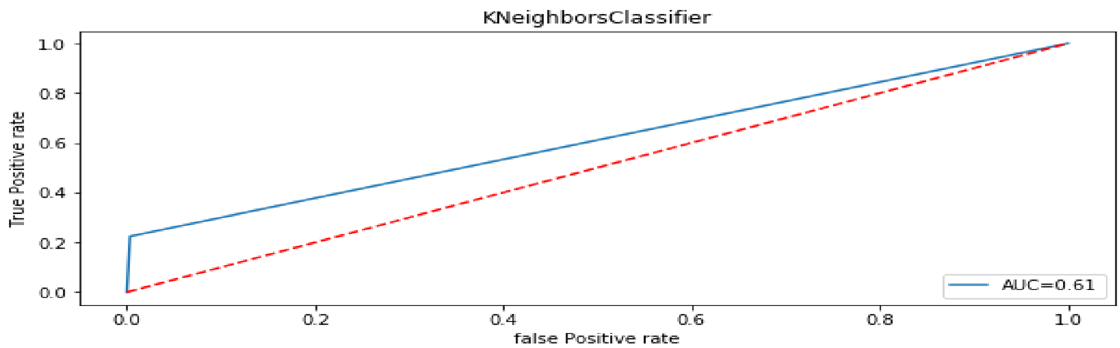
- Convert all texts into vectors
- Import classifier
- Train and test

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.metrics import classification_report,confusion_matrix,accuracy_score,roc_curve,auc
#Prediction - Classification Algorithms
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC, LinearSVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import cross_val_score
```

- **Run and Evaluate selected models**
    1. **kneighborsclassifier algorithm**

```
KNN=KNeighborsClassifier(n_neighbors=6)
Model.append('KNeighborsClassifier')
KNN.fit(x_train,y_train)
print(KNN)
pre=KNN.predict(x_test)
AS=accuracy_score(y_test,pre)
print('Accuracy_score= ',AS)
score.append(AS*100)
sc=cross_val_score(KNN,x,y,cv=5,scoring='accuracy').mean()
print('Cross_val_score=',sc)
cvs.append(sc*100)
false_positive_rate,true_positive_rate,thresholds=roc_curve(y_test,pre)
roc_auc=auc(false_positive_rate,true_positive_rate)
print('roc_auc_score= ',roc_auc)
rocscore.append(roc_auc*100)
print('classification_report\n',classification_report(y_test,pre),'\n')
cm=confusion_matrix(y_test,pre)
print('Confusion Matrix\n',cm,'\n')
plt.figure(figsize=(10,40))
plt.subplot(911)
plt.title('KNeighborsClassifier')
plt.plot(false_positive_rate,true_positive_rate,label='AUC=%0.2f'%roc_auc)
plt.plot([0,1],[0,1],'r--')
plt.legend(loc='lower right')
plt.ylabel('True Positive rate')
plt.xlabel('false Positive rate')
print('\n\n')
```

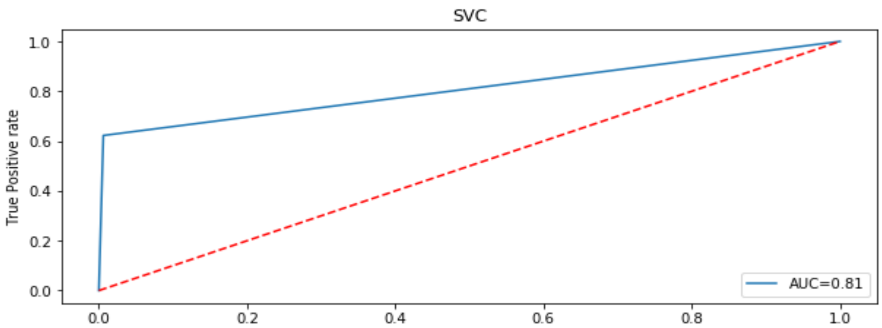
```
KNeighborsClassifier(n_neighbors=6)
Accuracy_score=  0.9172376336898396
Cross_val_score= 0.9177983431914599
roc_auc_score=  0.6105631081187836
classification_report
               precision    recall  f1-score   support

           0       0.92      1.00      0.96     42950
           1       0.88      0.22      0.36      4922

    accuracy                           0.92     47872
   macro avg       0.90      0.61      0.66     47872
weighted avg       0.91      0.92      0.89     47872


Confusion Matrix
 [[42805   145]
 [ 3817  1105]]
```



2. svc algorithm

```
SV=SVC()
Model.append('SVC')
SV.fit(x_train,y_train)
print(SV)
pre=SV.predict(x_test)
print('\n')
AS=accuracy_score(y_test,pre)
print('Accuracy_score= ',AS)
score.append(AS*100)
sc=cross_val_score(SV,x,y,cv=5,scoring='accuracy').mean()
print('Cross_val_score=',sc,'\n')
cvs.append(sc*100)
print('\n')
false_positive_rate,true_positive_rate,thresholds=roc_curve(y_test,pre)
roc_auc=auc(false_positive_rate,true_positive_rate)
print('roc_auc_score= ',roc_auc,'\n')
rocscore.append(roc_auc*100)
print('classification_report\n',classification_report(y_test,pre),'\n')
cm=confusion_matrix(y_test,pre)
print('Confusion Matrix\n',cm,'\n')
plt.figure(figsize=(10,40))
plt.subplot(911)
plt.title('SVC')
plt.plot(false_positive_rate,true_positive_rate,label='AUC=%0.2f'%roc_auc)
plt.plot([0,1],[0,1],'r--')
plt.legend(loc='lower right')
plt.ylabel('True Positive rate')
plt.xlabel('false Positive rate')
print('\n\n')
```

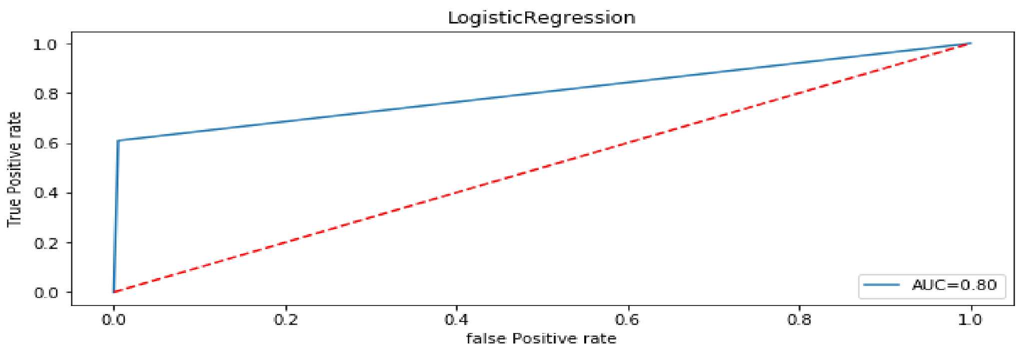
```
SVC()


Accuracy_score=  0.9557779077540107
Cross_val_score= 0.9569345282688333



roc_auc_score=  0.8082404532830905

classification_report
               precision    recall  f1-score   support

           0       0.96      0.99      0.98     42950
           1       0.92      0.62      0.74      4922

    accuracy                           0.96     47872
   macro avg       0.94      0.81      0.86     47872
weighted avg       0.95      0.96      0.95     47872


Confusion Matrix
 [[42691   259]
 [ 1858  3064]]
```



3. Logistic regression

```
LR=LogisticRegression()
Model.append('LogisticRegression')
LR.fit(x_train,y_train)
print(LR)
pre=LR.predict(x_test)
print('\n')
AS=accuracy_score(y_test,pre)
print('Accuracy_score= ',AS)
score.append(AS*100)
sc=cross_val_score(LR,x,y,cv=5,scoring='accuracy').mean()
print('Cross_val_score=',sc,'\n')
cvs.append(sc*100)
#print('\n')
false_positive_rate,true_positive_rate,thresholds=roc_curve(y_test,pre)
roc_auc=auc(false_positive_rate,true_positive_rate)
print('roc_auc_score= ',roc_auc,'\n')
rocscore.append(roc_auc*100)
print('classification_report\n',classification_report(y_test,pre),'\n')
cm=confusion_matrix(y_test,pre)
print('Confusion Matrix\n',cm,'\n')
plt.figure(figsize=(10,40))
plt.subplot(911)
plt.title('LogisticRegression')
plt.plot(false_positive_rate,true_positive_rate,label='AUC=%0.2f'%roc_auc)
plt.plot([0,1],[0,1],'r--')
plt.legend(loc='lower right')
plt.ylabel('True Positive rate')
plt.xlabel('false Positive rate')
print('\n\n')
```

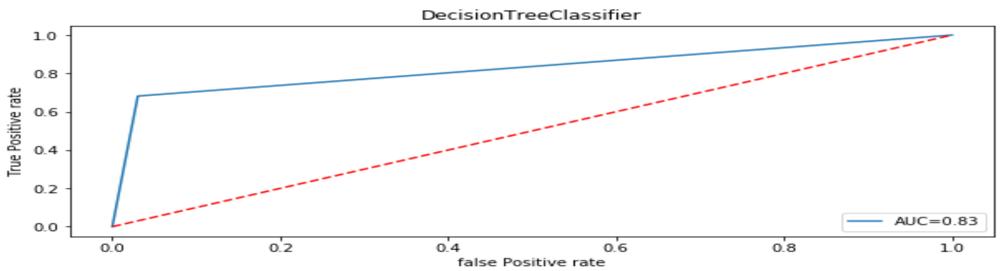
```
LogisticRegression()


Accuracy_score=  0.9553183489304813
Cross_val_score= 0.9556373011751551

roc_auc_score=  0.8018681986131497

classification_report
               precision    recall  f1-score   support

           0       0.96      1.00      0.98     42950
           1       0.93      0.61      0.74      4922

    accuracy                           0.96     47872
   macro avg       0.95      0.80      0.86     47872
weighted avg       0.95      0.96      0.95     47872


Confusion Matrix
 [[42737   213]
 [ 1926  2996]]
```



4. Decision tree classifier

```
DT=DecisionTreeClassifier(random_state=6)
Model.append('DecisionTreeClassifier')
DT.fit(x_train,y_train)
print(DT)
pre=DT.predict(x_test)
print('\n')
AS=accuracy_score(y_test,pre)
print('Accuracy_score= ',AS)
score.append(AS*100)
sc=cross_val_score(DT,x,y,cv=5,scoring='accuracy').mean()
print('Cross_val_score=',sc,'\n')
cvs.append(sc*100)
print('\n')
false_positive_rate,true_positive_rate,thresholds=roc_curve(y_test,pre)
roc_auc=auc(false_positive_rate,true_positive_rate)
print('roc_auc_score= ',roc_auc,'\n')
rocscore.append(roc_auc*100)
print('classification_report\n',classification_report(y_test,pre),'\n')
cm=confusion_matrix(y_test,pre)
print('Confusion Matrix\n',cm,'\n')
plt.figure(figsize=(10,40))
plt.subplot(911)
plt.title('DecisionTreeClassifier')
plt.plot(false_positive_rate,true_positive_rate,label='AUC=%0.2f'%roc_auc)
plt.plot([0,1],[0,1],'r--')
plt.legend(loc='lower right')
plt.ylabel('True Positive rate')
plt.xlabel('false Positive rate')
print('\n\n')
```

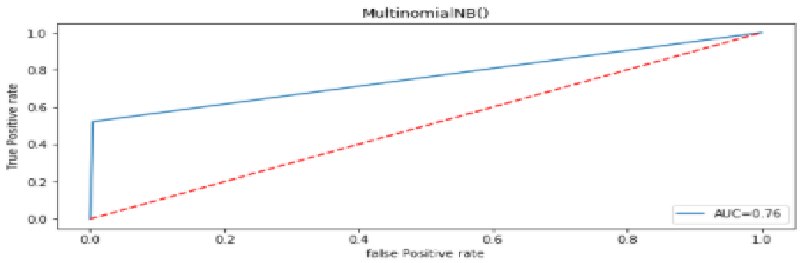
```
DecisionTreeClassifier(random_state=6)

Accuracy_score=  0.9400066844919787
Cross_val_score= 0.9419568656878473


roc_auc_score=  0.8257146006218546

classification_report
               precision    recall  f1-score   support

           0       0.96      0.97      0.97     42950
           1       0.72      0.68      0.70      4922

    accuracy                           0.94     47872
   macro avg       0.84      0.83      0.83     47872
weighted avg       0.94      0.94      0.94     47872


Confusion Matrix
 [[41644  1306]
 [ 1566  3356]]
```



5. MultinomialNB

```
native =MultinomialNB()
Model.append('MultinomialNB')
native.fit(x_train,y_train)
print(native)
pre=native.predict(x_test)
print('\n')
AS=accuracy_score(y_test,pre)
print('Accuracy_score= ',AS)
score.append(AS*100)
sc=cross_val_score(native,x,y,cv=5,scoring='accuracy').mean()
print('Cross_val_score=',sc,'\n')
cvs.append(sc*100)
print('\n')
false_positive_rate,true_positive_rate,thresholds=roc_curve(y_test,pre)
roc_auc=auc(false_positive_rate,true_positive_rate)
print('roc_auc_score= ',roc_auc,'\n')
rocscore.append(roc_auc*100)
print('classification_report\n',classification_report(y_test,pre),'\n')
cm=confusion_matrix(y_test,pre)
print('Confusion Matrix\n',cm,'\n')
plt.figure(figsize=(10,40))
plt.subplot(911)
plt.title(native)
plt.plot(false_positive_rate,true_positive_rate,label='AUC=%0.2f'%roc_auc)
plt.plot([0,1],[0,1],'r--')
plt.legend(loc='lower right')
plt.ylabel('True Positive rate')
plt.xlabel('false Positive rate')
print('\n\n')
```

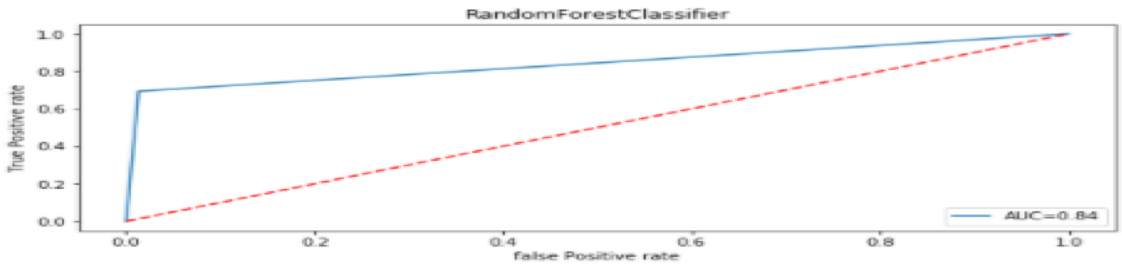
```
MultinomialNB()


Accuracy_score=  0.9473387366310161
Cross_val_score= 0.9484994139280257


roc_auc_score=  0.7588355008682597

classification_report
               precision    recall  f1-score   support

           0       0.95      1.00      0.97     42950
           1       0.94      0.52      0.67      4922

    accuracy                           0.95     47872
   macro avg       0.94      0.76      0.82     47872
weighted avg       0.95      0.95      0.94     47872


Confusion Matrix
 [[42784   166]
 [ 2355  2567]]
```



6. RandomForestClassifier

```
RFC=RandomForestClassifier(n_estimators=1000,random_state=0)
Model.append('RandomForestClassifier')
RFC.fit(x_train,y_train)
print(RFC)
pre=RFC.predict(x_test)
print('\n')
AS=accuracy_score(y_test,pre)
print('Accuracy_score= ',AS)
score.append(AS*100)
sc=cross_val_score(RFC,x,y,cv=5,scoring='accuracy').mean()
print('Cross_val_score=',sc,'\n')
cvs.append(sc*100)
print('\n')
false_positive_rate,true_positive_rate,thresholds=roc_curve(y_test,pre)
roc_auc=auc(false_positive_rate,true_positive_rate)
print('roc_auc_score= ',roc_auc,'\n')
rocscore.append(roc_auc*100)
print('classification_report\n',classification_report(y_test,pre),'\n')
cm=confusion_matrix(y_test,pre)
print('Confusion Matrix\n',cm,'\n')
plt.figure(figsize=(10,40))
plt.subplot(911)
plt.title('RandomForestClassifier')
plt.plot(false_positive_rate,true_positive_rate,label='AUC=%0.2f'%roc_auc)
plt.plot([0,1],[0,1],'r--')
plt.legend(loc='lower right')
plt.ylabel('True Positive rate')
plt.xlabel('false Positive rate')
print('\n\n')
```

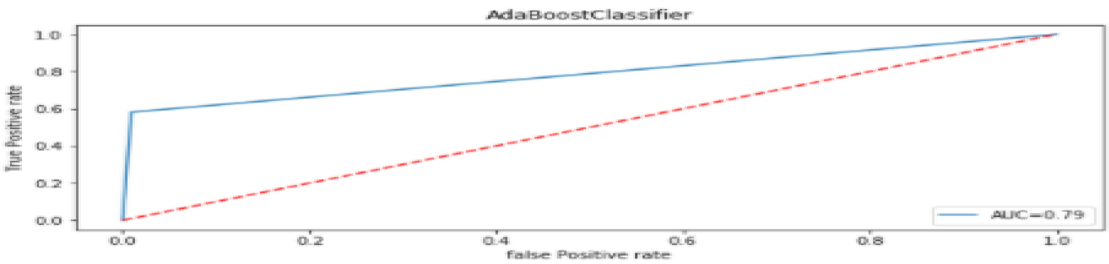
```
RandomForestClassifier(n_estimators=1000, random_state=0)


Accuracy_score=  0.956613469251337
Cross_val_score= 0.957041063799742



roc_auc_score=  0.840096315088134

classification_report
              precision    recall  f1-score   support

           0       0.97      0.99      0.98     42950
           1       0.86      0.69      0.77      4922

    accuracy                           0.96     47872
   macro avg       0.91      0.84      0.87     47872
weighted avg       0.95      0.96      0.95     47872


Confusion Matrix
 [[42382   568]
 [ 1509  3413]]
```



7.AdaBoostClassifier

```
from sklearn.ensemble import AdaBoostClassifier,GradientBoostingClassifier
ada=AdaBoostClassifier(n_estimators=100)
Model.append('AdaBoostClassifier')
ada.fit(x_train,y_train)
print(ada)
pre=ada.predict(x_test)
print('\n')
AS=accuracy_score(y_test,pre)
print('Accuracy_score= ',AS)
score.append(AS*100)
sc=cross_val_score(DT,x,y,cv=5,scoring='accuracy').mean()
print('Cross_val_score=',sc,'\n')
cvs.append(sc*100)
print('\n')
false_positive_rate,true_positive_rate,thresholds=roc_curve(y_test,pre)
roc_auc=auc(false_positive_rate,true_positive_rate)
print('roc_auc_score= ',roc_auc,'\n')
rocscore.append(roc_auc*100)
print('classification_report\n',classification_report(y_test,pre),'\n')
cm=confusion_matrix(y_test,pre)
print('Confusion Matrix\n',cm,'\n')
plt.figure(figsize=(10,40))
plt.subplot(911)
plt.title('AdaBoostClassifier')
plt.plot(false_positive_rate,true_positive_rate,label='AUC=%0.2f'%roc_auc)
plt.plot([0,1],[0,1],'r--')
plt.legend(loc='lower right')
plt.ylabel('True Positive rate')
plt.xlabel('false Positive rate')
print('\n\n')
```

```
AdaBoostClassifier(n_estimators=100)

Accuracy_score=  0.9486965240641712
Cross_val_score= 0.9419568656878473


roc_auc_score=  0.7860355184652404

classification_report
               precision    recall  f1-score   support

           0       0.95      0.99      0.97     42950
           1       0.88      0.58      0.70      4922

    accuracy                           0.95     47872
   macro avg       0.92      0.79      0.84     47872
weighted avg       0.95      0.95      0.94     47872


Confusion Matrix
 [[42555   395]
 [ 2061  2861]]
```



- **Final Result of models**

```
result
```

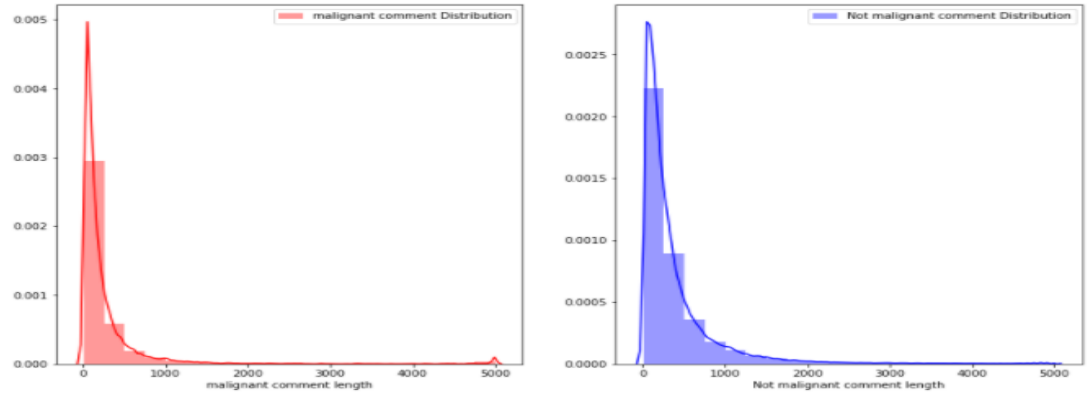
	Model	Accuracy_score	Cross_val_score	Roc_auc_curve
0	KNeighborsClassifier	91.7238	91.7798	61.0563
1	SVC	95.5778	95.6935	80.824
2	LogisticRegression	95.5318	95.5637	80.1868
3	DecisionTreeClassifier	94.0007	94.1957	82.5715
4	MultinomialNB	94.7339	94.8499	75.8836
5	RandomForestClassifier	95.6613	95.7041	84.0096
7	AdaBoostClassifier	94.8697	94.1957	78.6036

```
#from sklearn.externals
import joblib
joblib.dump(RFC,'Malignant_Comment_Classifier_Model.pkl')
```

- **Visualizations**

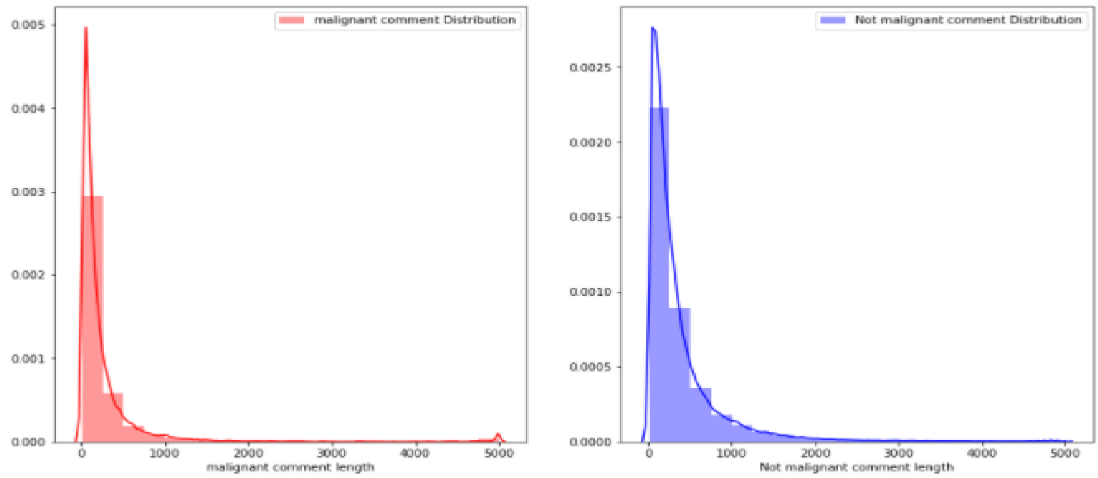
1.

```
#comment distribution Before cleaning
f,ax =plt.subplots(1,2,figsize=(15,8))
sns.distplot(df_train[df_train['malignant']==1]['length'],bins=20,ax=ax[0],label='malignant comment Distribution',color='r')
ax[0].set_xlabel('malignant comment length')
ax[0].legend()
sns.distplot(df_train[df_train['malignant']==0]['length'],bins=20,ax=ax[1],label='Not malignant comment Distribution',color='b')
ax[1].set_xlabel('Not malignant comment length')
ax[1].legend()
plt.show()
```



2.

```
#comment distribution Before cleaning
f,ax =plt.subplots(1,2,figsize=(15,8))
sns.distplot(df_train[df_train['malignant']==1]['length'],bins=20,ax=ax[0],label='malignant comment Distribution',color='r')
ax[0].set_xlabel('malignant comment length')
ax[0].legend()
sns.distplot(df_train[df_train['malignant']==0]['length'],bins=20,ax=ax[1],label='Not malignant comment Distribution',color='b')
ax[1].set_xlabel('Not malignant comment length')
ax[1].legend()
plt.show()
```

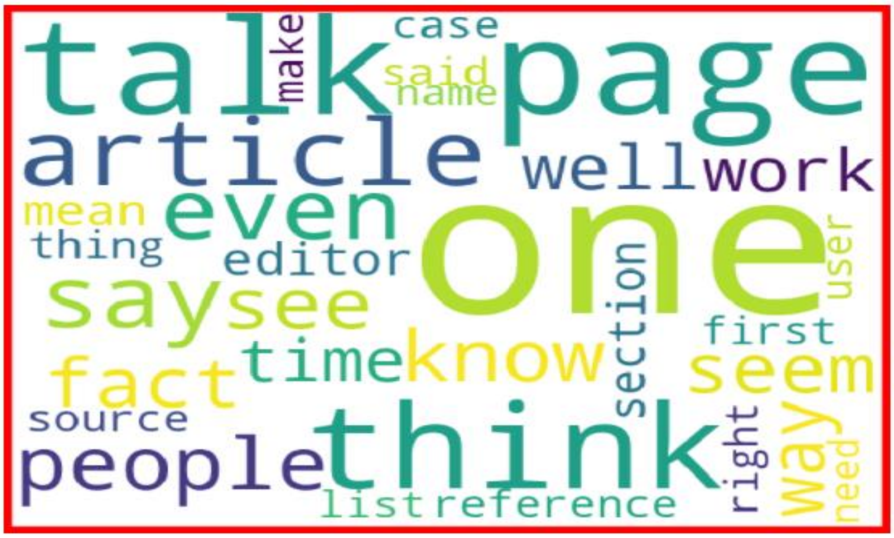


3.

```
#getting sense of loud word in malignant
from wordcloud import WordCloud

malignant=df_train['comment_text'][df_train['malignant']==1]

malignant_cloud=WordCloud(width=700,height=500,background_color='white',max_words=30).generate(' '.join(malignant))
plt.figure(figsize=(10,8),facecolor='r')
plt.imshow(malignant_cloud)
plt.axis('off')
plt.tight_layout(pad=0)
plt.show()
```

4.

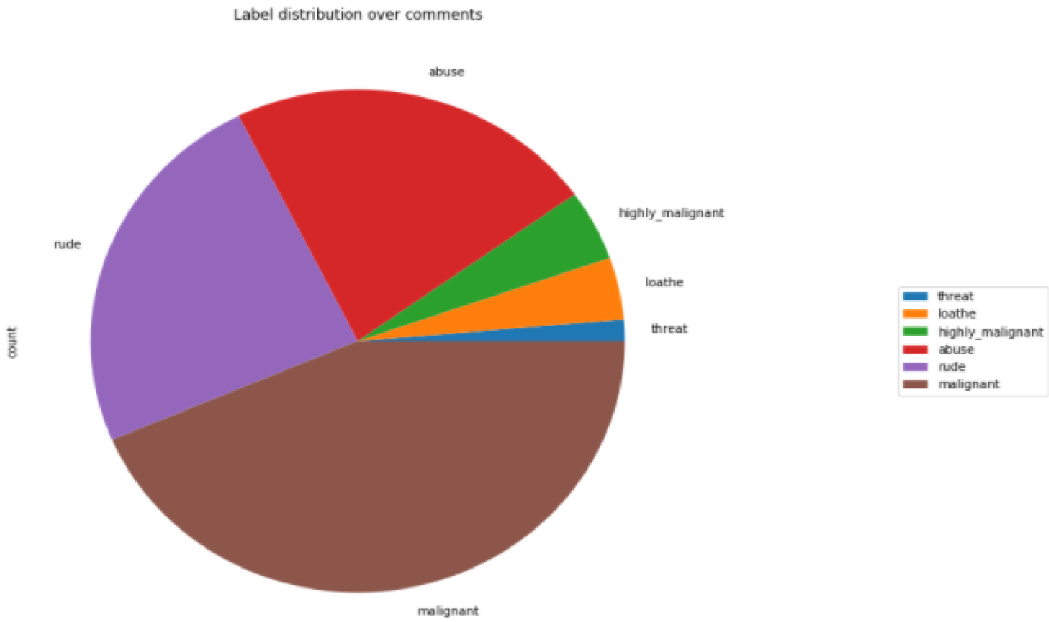
```
#getting sense of loud word in Not malignant
from wordcloud import WordCloud

not_malignant=df_train['comment_text'][df_train['malignant']==0]

not_malignant_cloud=WordCloud(width=700,height=500,background_color='white',max_words=30).generate(' '.join(not_malignant))
plt.figure(figsize=(10,8),facecolor='r')
plt.imshow(not_malignant_cloud)
plt.axis('off')
plt.tight_layout(pad=0)
plt.show()
```



4.

```
cols_target = ['malignant','highly_malignant','rude','threat','abuse','loathe']
df_distribution =df_train[cols_target].sum()\
                            .to_frame()\
                            .rename(columns={0: 'count'})\
                            .sort_values('count')

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

```
<matplotlib.legend.Legend at 0x2b1ce0e1c08>
```



CONCLUSION

After analysing data, visualization, and modelling, we conclude that using the Random Forest Classifier is suitable for modelling of comment's category prediction.