

Malignant Comment Classification

Submitted by:

Priyanka Yadav

# ACKNOWLEDGMENT

I would like to thank FlipRobo Technologies to provide me this valuable data in order to perform this analysis. I would also like to thank my mentor Swati Mahaseth for helping me throughout this project.

##Importing all necessary libraries

import numpy as np

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

train=pd.read\_csv('../input/malignant-comment-classification/ train.csv')

train.head()

id comment\_text

\

0000997932d777bf Explanation\nWhy the edits made under my usern...

000103f0d9cfb60f D'aww! He matches this background colour I'm s...

000113f07ec002fd Hey man, I'm really not trying to edit war. It...

0001b41b1c6bb37e "\nMore\nI can't make any real suggestions on ...

0001d958c54c6e35 You, sir, are my hero. Any chance you remember...

malignant highly\_malignant rude threat abuse loathe 0 0 0 0 0 0 0

1 0 0 0 0 0 0

2 0 0 0 0 0 0

3 0 0 0 0 0 0

4 0 0 0 0 0 0

test=pd.read\_csv('../input/malignant-comment-classification/test.csv') test.head()

id comment\_text

00001cee341fdb12 Yo bitch Ja Rule is more succesful then you'll...

0000247867823ef7 == From RfC == \n\n The title is fine as it is...

00013b17ad220c46 " \n\n == Sources == \n\n \* Zawe Ashton on Lap...

00017563c3f7919a :If you have a look back at the source, the in...

00017695ad8997eb I don't anonymously edit articles at all.

print('train shape is ',train.shape) print('test shape is ',test.shape) print('test info',test.info)

print('train info',train.info)

train shape is (159571, 8)

test shape is (153164, 2)

test info <bound method DataFrame.info of id comment\_text

00001cee341fdb12 Yo bitch Ja Rule is more succesful then you'll...

0000247867823ef7 == From RfC == \n\n The title is fine as it is...

00013b17ad220c46 " \n\n == Sources == \n\n \* Zawe Ashton on Lap...

00017563c3f7919a :If you have a look back at the source, the in...

00017695ad8997eb I don't anonymously edit articles at all.

... ...

...

153159 fffcd0960ee309b5 . \n i totally agree, this stuff is nothing bu...

153160 fffd7a9a6eb32c16 == Throw from out field to home plate. == \ n\n...

153161 fffda9e8d6fafa9e " \n\n == Okinotorishima categories == \n\n I ...

153162 fffe8f1340a79fc2 " \n\n == ""One of the founding nations of the...

153163 ffffce3fb183ee80 " \n :::Stop already. Your bullshit is not wel...

[153164 rows x 2 columns]>

train info <bound method DataFrame.info of id comment\_text \

0000997932d777bf Explanation\nWhy the edits made under my usern...

000103f0d9cfb60f D'aww! He matches this background colour I'm s...

000113f07ec002fd Hey man, I'm really not trying to edit war. It...

0001b41b1c6bb37e "\nMore\nI can't make any real suggestions on ...

0001d958c54c6e35 You, sir, are my hero. Any chance you remember...

... ...

...

159566 ffe987279560d7ff ":::::And for the second time of asking, when ...

159567 ffea4adeee384e90 You should be ashamed of yourself \n\nThat is ...

159568 ffee36eab5c267c9 Spitzer \n\nUmm, theres no actual article for ...

159569 fff125370e4aaaf3 And it looks like it was actually you who put ...

159570 fff46fc426af1f9a "\nAnd ... I really don't think you understand...

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | malignant | highly\_malignant | rude | threat | abuse | loathe |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4 | 0 | 0 | 0 | 0 | 0 | 0 |
| ... | ... | ... | ... | ... | ... | ... |
| 159566 | 0 | 0 | 0 | 0 | 0 | 0 |
| 159567 | 0 | 0 | 0 | 0 | 0 | 0 |
| 159568 | 0 | 0 | 0 | 0 | 0 | 0 |
| 159569 | 0 | 0 | 0 | 0 | 0 | 0 |
| 159570 | 0 | 0 | 0 | 0 | 0 | 0 |

[159571 rows x 8 columns]>

print('train data Set descriptin',train.describe()) print('test data Set descriptin',test.describe())

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| train rude | data Set descriptin  threat \ | | malignant | | highly\_malignant | |
| count | 159571.000000 | | 159571.000000 | 159571.000000 | | 159571.000000 |
| mean | 0.095844 | | 0.009996 | 0.052948 | | 0.002996 |
| std | 0.294379 | | 0.099477 | 0.223931 | | 0.054650 |
| min | 0.000000 | | 0.000000 | 0.000000 | | 0.000000 |
| 25% | 0.000000 | | 0.000000 | 0.000000 | | 0.000000 |
| 50% | 0.000000 | | 0.000000 | 0.000000 | | 0.000000 |
| 75% | 0.000000 | | 0.000000 | 0.000000 | | 0.000000 |
| max | 1.000000 | | 1.000000 | 1.000000 | | 1.000000 |
|  | abuse | loathe | | | | |
| count | 159571.000000 | 159571.000000 | | | | |
| mean | 0.049364 | 0.008805 | | | | |
| std | 0.216627 | 0.093420 | | | | |
| min | 0.000000 | 0.000000 | | | | |
| 25% | 0.000000 | 0.000000 | | | | |
| 50% | 0.000000 | 0.000000 | | | | |
| 75% | 0.000000 | 0.000000 | | | | |
| max | 1.000000 | 1.000000 | | | | |

test data Set descriptin id comment\_text

count 153164

153164

unique 153164

153164

top 821fdc095707837b Sockpuppet== \n See Wikipedia:Requests for com...

freq 1

1

# checking null values print(train.isnull().sum()) print(sns.heatmap(train.isnull()))

id 0

comment\_text 0

malignant 0

highly\_malignant 0

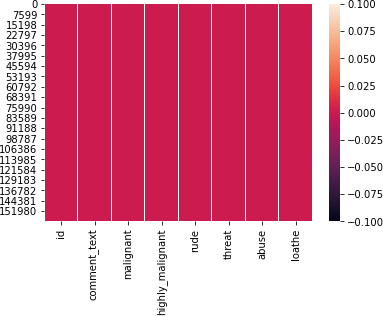
rude 0

threat 0

abuse 0

loathe 0

dtype: int64 AxesSubplot(0.125,0.125;0.62x0.755)



## checking correlation in dataset print(train.corr()) print(sns.heatmap(train.corr()))

|  |  |  |  |
| --- | --- | --- | --- |
| malignant | highly\_malignant | rude | threat |
| 1.000000 | 0.308619 | 0.676515 | 0.157058 |
| 0.308619 | 1.000000 | 0.403014 | 0.123601 |
| 0.676515 | 0.403014 | 1.000000 | 0.141179 |
| 0.157058 | 0.123601 | 0.141179 | 1.000000 |
| 0.647518 | 0.375807 | 0.741272 | 0.150022 |
| 0.266009 | 0.201600 | 0.286867 | 0.115128 |
| loathe |  |  |  |

abuse \ malignant 0.647518

highly\_malignant 0.375807

rude 0.741272

threat 0.150022

abuse 1.000000

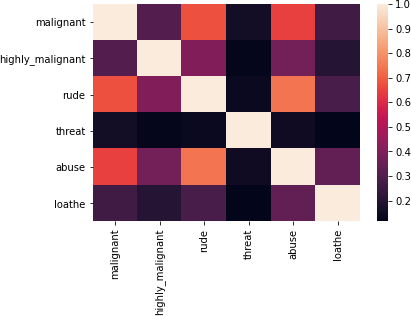
loathe 0.337736

malignant 0.266009

highly\_malignant 0.201600

|  |  |
| --- | --- |
| rude | 0.286867 |
| threat | 0.115128 |
| abuse | 0.337736 |
| loathe | 1.000000 |

AxesSubplot(0.125,0.125;0.62x0.755)



# checking the skewness for the features:

train.skew()

|  |  |
| --- | --- |
| malignant | 2.745854 |
| highly\_malignant | 9.851722 |
| rude | 3.992817 |
| threat | 18.189001 |
| abuse | 4.160540 |
| loathe | 10.515923 |
| dtype: float64 |  |

col=['malignant','highly\_malignant','loathe','rude','abuse','threat']

**for** i **in** col:

print(i) print("\n")

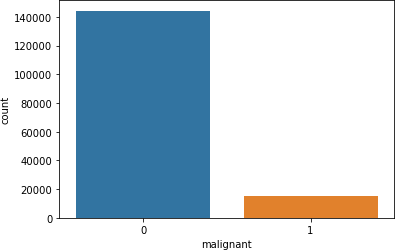
print(train[i].value\_counts()) sns.countplot(train[i]) plt.show()

malignant

0 144277

1 15294

Name: malignant, dtype: int64

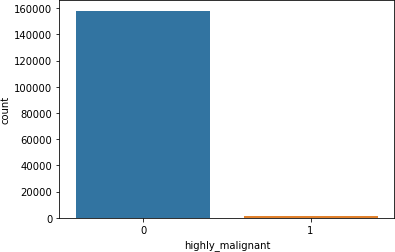


highly\_malignant

0 157976

1 1595

Name: highly\_malignant, dtype: int64

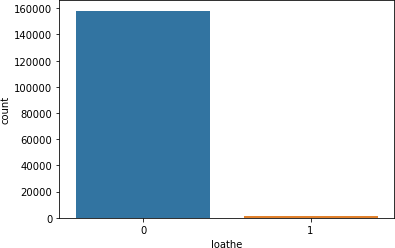


loathe

0 158166

1 1405

Name: loathe, dtype: int64

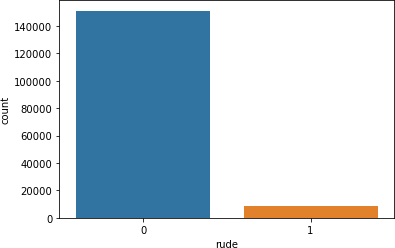


rude

0 151122

1 8449

Name: rude, dtype: int64

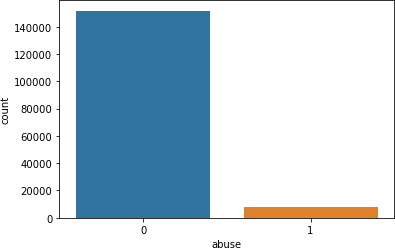


abuse

0 151694

1 7877

Name: abuse, dtype: int64

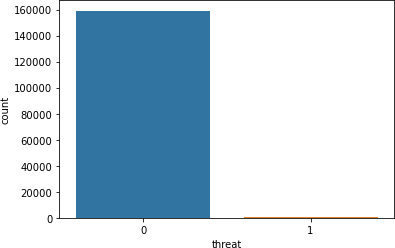


threat

0 159093

1 478

Name: threat, dtype: int64



from nltk.stem import WordNetLemmatizer import nltk

from nltk.corpus import stopwords import string

train['length'] = train['comment\_text'].str.len() train.head(2)

id comment\_text

\

0000997932d777bf Explanation\nWhy the edits made under my usern...

000103f0d9cfb60f D'aww! He matches this background colour I'm s...

malignant highly\_malignant rude threat abuse loathe length

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 264 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 112 |

# Convert all messages to lower case

train['comment\_text'] = train['comment\_text'].str.lower()

# Replace email addresses with 'email'

train['comment\_text'] = train['comment\_text'].str.replace(r'^.

+@[^\.].\*\.[a-z]{2,}$',

'emailaddress')

# Replace URLs with 'webaddress'

train['comment\_text'] = train['comment\_text'].str.replace(r'^http\://[a-zA-Z0-9\-\.]+\.[a-zA- Z]{2,3}(/\S\*)?$',

'webaddress')

# Replace money symbols with 'moneysymb' (£ can by typed with ALT key

+ 156)

train['comment\_text'] = train['comment\_text'].str.replace(r'£|\$', 'dollers')

# Replace 10 digit phone numbers (formats include paranthesis, spaces, no spaces, dashes) with 'phonenumber'

train['comment\_text'] = train['comment\_text'].str.replace(r'^\(?[\d]

{3}\)?[\s-]?[\d]{3}[\s-]?[\d]{4}$',

'phonenumber')

# Replace numbers with 'numbr'

train['comment\_text'] = train['comment\_text'].str.replace(r'\d+(\.\ d+)?', 'numbr')

train['comment\_text'] = train['comment\_text'].apply(**lambda** x: ' '.join(

term **for** term **in** x.split() **if** term **not in** string.punctuation))

stop\_words = set(stopwords.words('english') + ['u', 'ü', 'ur', '4',

'2', 'im', 'dont', 'doin', 'ure'])

train['comment\_text'] = train['comment\_text'].apply(**lambda** x: ' '.join(

term **for** term **in** x.split() **if** term **not in** stop\_words))

lem=WordNetLemmatizer()

train['comment\_text'] = train['comment\_text'].apply(**lambda** x: ' '.join(

lem.lemmatize(t) **for** t **in** x.split()))

train['clean\_length'] = train.comment\_text.str.len() train.head()

# Total length removal

print ('Origian Length', train.length.sum()) print ('Clean Length', train.clean\_length.sum())

#Getting sense of loud words which are offensive

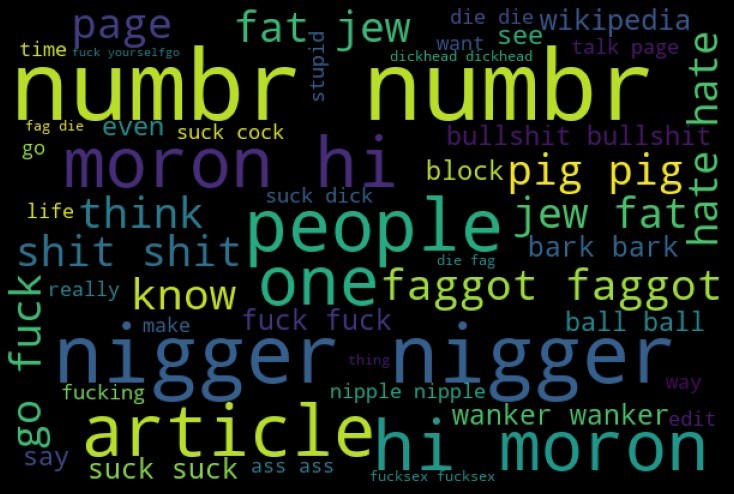
from wordcloud import WordCloud

hams = train['comment\_text'][train['malignant']==1] spam\_cloud =

WordCloud(width=600,height=400,background\_color='black',max\_words=50). generate(' '.join(hams))

plt.figure(figsize=(10,8),facecolor='k') plt.imshow(spam\_cloud)

plt.axis('off') plt.tight\_layout(pad=0) plt.show()



from sklearn.naive\_bayes import MultinomialNB

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import accuracy\_score, confusion\_matrix, classification\_report,roc\_curve,roc\_auc\_score,auc

from sklearn.model\_selection import train\_test\_split from sklearn.metrics import

accuracy\_score,classification\_report,confusion\_matrix,f1\_score from sklearn.linear\_model import LogisticRegression

from sklearn.model\_selection import cross\_val\_score,GridSearchCV from sklearn.naive\_bayes import MultinomialNB

from sklearn.tree import DecisionTreeClassifier from sklearn.neighbors import KNeighborsClassifier from sklearn.ensemble import

RandomForestClassifier,AdaBoostClassifier,GradientBoostingClassifier from sklearn.naive\_bayes import GaussianNB

from sklearn.linear\_model import LogisticRegression from sklearn.svm import SVC

from sklearn.tree import DecisionTreeClassifier

cols\_target = ['malignant','highly\_malignant','rude','threat','abuse','loathe'] df\_distribution = train[cols\_target].sum()\

.to\_frame()\

.rename(columns={0: 'count'})\

.sort\_values('count')

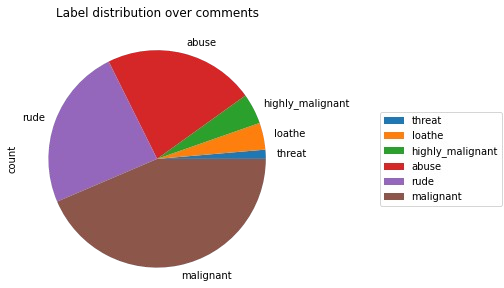
df\_distribution.plot.pie(y='count', comments',

title='Label distribution over figsize=(5, 5))\

.legend(loc='center left',

bbox\_to\_anchor=(1.3, 0.5))

<matplotlib.legend.Legend at 0x7fd6b3783a50>



target\_data = train[cols\_target]

train['bad'] =train[cols\_target].sum(axis =1) print(train['bad'].value\_counts()) train['bad'] = train['bad'] > 0

train['bad'] = train['bad'].astype(int) print(train['bad'].value\_counts())

|  |  |
| --- | --- |
| 0 | 143346 |
| 1 | 6360 |
| 3 | 4209 |
| 2 | 3480 |
| 4 | 1760 |
| 5 | 385 |
| 6 | 31 |

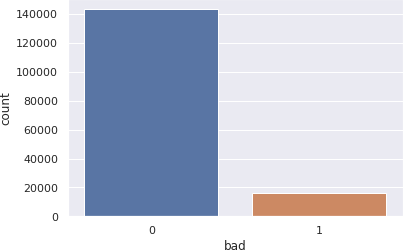
Name: bad, dtype: int64 0 143346

1 16225

Name: bad, dtype: int64

sns.set()

sns.countplot(x="bad" , data = train) plt.show()



# 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(train['comment\_text'])

x = features train.shape (159571, 11)

test.shape (153164, 2)

y=train['bad'] x\_train,x\_test,y\_train,y\_test=train\_test\_split(x,y,random\_state=56,tes t\_size=.30)

y\_train.shape,y\_test.shape ((111699,), (47872,))

# LogisticRegression

LG = LogisticRegression(C=1, max\_iter = 3000)

LG.fit(x\_train, y\_train)

y\_pred\_train = LG.predict(x\_train)

print('Training accuracy is {}'.format(accuracy\_score(y\_train, y\_pred\_train)))

y\_pred\_test = LG.predict(x\_test) print('Test accuracy is

{}'.format(accuracy\_score(y\_test,y\_pred\_test))) print(confusion\_matrix(y\_test,y\_pred\_test)) print(classification\_report(y\_test,y\_pred\_test))

Training accuracy is 0.9595520103134316 Test accuracy is 0.9552974598930482 [[42729 221]

|  |  |  |  |
| --- | --- | --- | --- |
| [ 1919 3003]]  precision | recall | f1-score | support |
| 0 0.96 | 0.99 | 0.98 | 42950 |
| 1 0.93 | 0.61 | 0.74 | 4922 |
| accuracy |  | 0.96 | 47872 |
| macro avg 0.94 | 0.80 | 0.86 | 47872 |
| weighted avg 0.95 | 0.96 | 0.95 | 47872 |

# DecisionTreeClassifier

DT = DecisionTreeClassifier()

DT.fit(x\_train, y\_train) y\_pred\_train = DT.predict(x\_train)

print('Training accuracy is {}'.format(accuracy\_score(y\_train, y\_pred\_train)))

y\_pred\_test = DT.predict(x\_test) print('Test accuracy is

{}'.format(accuracy\_score(y\_test,y\_pred\_test))) print(confusion\_matrix(y\_test,y\_pred\_test)) print(classification\_report(y\_test,y\_pred\_test))

Training accuracy is 0.9988898736783678 Test accuracy is 0.9391920120320856 [[41580 1370]

[ 1541 3381]]

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | precision | recall | f1-score | support |
| 0 | 0.96 | 0.97 | 0.97 | 42950 |
| 1 | 0.71 | 0.69 | 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

#RandomForestClassifier

RF = RandomForestClassifier()

RF.fit(x\_train, y\_train) y\_pred\_train = RF.predict(x\_train)

print('Training accuracy is {}'.format(accuracy\_score(y\_train, y\_pred\_train)))

y\_pred\_test = RF.predict(x\_test) print('Test accuracy is

{}'.format(accuracy\_score(y\_test,y\_pred\_test))) print(confusion\_matrix(y\_test,y\_pred\_test)) print(classification\_report(y\_test,y\_pred\_test))

Training accuracy is 0.9988809210467416 Test accuracy is 0.9546707887700535 [[42383 567]

|  |  |  |  |
| --- | --- | --- | --- |
| [ 1603 3319]]  precision | recall | f1-score | support |
| 0 0.96 | 0.99 | 0.98 | 42950 |
| 1 0.85 | 0.67 | 0.75 | 4922 |
| accuracy |  | 0.95 | 47872 |
| macro avg 0.91 | 0.83 | 0.86 | 47872 |
| weighted avg 0.95 | 0.95 | 0.95 | 47872 |

# xgboost

import xgboost

xgb = xgboost.XGBClassifier() xgb.fit(x\_train, y\_train) y\_pred\_train = xgb.predict(x\_train)

print('Training accuracy is {}'.format(accuracy\_score(y\_train, y\_pred\_train)))

y\_pred\_test = xgb.predict(x\_test) print('Test accuracy is

{}'.format(accuracy\_score(y\_test,y\_pred\_test))) print(confusion\_matrix(y\_test,y\_pred\_test)) print(classification\_report(y\_test,y\_pred\_test))

Training accuracy is 0.9614052050600274 Test accuracy is 0.9526236631016043 [[42689 261]

[ 2007 2915]]

precision recall f1-score support 0 0.96 0.99 0.97 42950

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 0.92 | 0.59 | 0.72 | 4922 |
| accuracy |  |  | 0.95 | 47872 |
| macro avg | 0.94 | 0.79 | 0.85 | 47872 |
| weighted avg | 0.95 | 0.95 | 0.95 | 47872 |

#AdaBoostClassifier ada=AdaBoostClassifier(n\_estimators=100) ada.fit(x\_train, y\_train)

y\_pred\_train = ada.predict(x\_train)

print('Training accuracy is {}'.format(accuracy\_score(y\_train, y\_pred\_train)))

y\_pred\_test = ada.predict(x\_test) print('Test accuracy is

{}'.format(accuracy\_score(y\_test,y\_pred\_test))) print(confusion\_matrix(y\_test,y\_pred\_test)) print(classification\_report(y\_test,y\_pred\_test))

Training accuracy is 0.951118631321677 Test accuracy is 0.9490307486631016 [[42553 397]

|  |  |  |  |
| --- | --- | --- | --- |
| [ 2043 2879]]  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 |

#KNeighborsClassifier knn=KNeighborsClassifier(n\_neighbors=9) knn.fit(x\_train, y\_train)

y\_pred\_train = knn.predict(x\_train)

print('Training accuracy is {}'.format(accuracy\_score(y\_train, y\_pred\_train)))

y\_pred\_test = knn.predict(x\_test) print('Test accuracy is

{}'.format(accuracy\_score(y\_test,y\_pred\_test))) print(confusion\_matrix(y\_test,y\_pred\_test)) print(classification\_report(y\_test,y\_pred\_test))

Training accuracy is 0.922300110117369 Test accuracy is 0.9173629679144385 [[42809 141]

[ 3815 1107]]

precision recall f1-score support

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0 | 0.92 | 1.00 | 0.96 | 42950 |
| 1 | 0.89 | 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 |

# RandomForestClassifier

RF = RandomForestClassifier() RF.fit(x\_train, y\_train) y\_pred\_train = RF.predict(x\_train)

print('Training accuracy is {}'.format(accuracy\_score(y\_train, y\_pred\_train)))

y\_pred\_test = RF.predict(x\_test) print('Test accuracy is

{}'.format(accuracy\_score(y\_test,y\_pred\_test))) cvs=cross\_val\_score(RF, x, y, cv=10, scoring='accuracy').mean() print('cross validation score :',cvs\*100) print(confusion\_matrix(y\_test,y\_pred\_test)) print(classification\_report(y\_test,y\_pred\_test))

Training accuracy is 0.9988898736783678 Test accuracy is 0.9556107954545454

#Plotting the graph which tells us about the area under curve , more the area under curve more will be the better prediction

# model is performing good : fpr,tpr,thresholds=roc\_curve(y\_test,y\_pred\_test) roc\_auc=auc(fpr,tpr)

plt.plot([0,1],[1,0],'k--')

plt.plot(fpr,tpr,label = 'RF Classifier') plt.xlabel('False positive rate') plt.ylabel('True positive rate') plt.title('RF CLASSIFIER')

plt.show()

import eli5

eli5.show\_weights(RF,vec = tf\_vec, top = 15) #random forest

# will give you top 15 features or words which makes a comment toxic

test\_data =tf\_vec.fit\_transform(test['comment\_text']) test\_data

prediction=RF.predict(test\_data) prediction

import joblib joblib.dump(RF,"malig.pkl")