“Malignant Comment Classification”

Submitted by:
MD RASHIDUNNABI

1. Problem Definition

1.**Project Overview**

Only if you think you can express yourself openly and without hesitation can Online platforms be utilized pleasantly by normal people. If a malignant or poisonous response, or any threat, or insult or harassment that makes them uncomfortable, is encountered, they may postpone to use the social media platform in future. Therefore, an automated system that can easily detect and track all these comments and therefore take any relevant action, for instance reporting or banning it, becomes incredibly important for any company or community to have in order to prevent similar problems in the future.

This is a major problem as there are 7.7 billion people in the globe and over 3.5 billion people utilize some kind of social media outlet, among those 7.7 billion. This indicates that each person utilizes a social networking platform one in three. As it is under the area of natural language processing, this difficulty may be removed. Here we are trying to understand the speaker's purpose by creating a model which detects several sorts of toxicities, such as threats, profanity, insults and hate depending on identification. Moreover, it is important that such an annoyance be handled and that a user-friendly experience is only possible afterwards.

2.**Problem Statement**

As a user comment on a number of remarks, phrases or paragraphs, our job is to identify a comment whether or not it's a cancerous comment. After this, our major duty is to classify tweets or comments into one or more of the following categories after we have all the malignant commentaries – toxic, really toxic, obscene, threatening, insulting, or identity-hating. The topic is therefore classified as a multilabel problem.

There is a distinction between the categorization of the traditional and renowned multiclases and that of the multi-label classification that we are utilizing. Each instance should be categorized into three or more classes within a multi-class classification, and for the same case a multi-label classification shall be anticipated on several labels (e.g. poisonous, very toxic, obscene, threatened, insulted, or hateful of identity).

This categorization challenge can be addressed in several ways. It is possible to use –

- → Category of multilabel processes: LP, Binary Relevance(BR), BR+ (BRplus), Classificator and Multi-label techniques which belong to the issue transformation category.

→ Base and algorithms adapted such as: J48 (Decision Tree), Naïve Bayes, k-Nearest-Neighbor (KNN), SMO (Support Vector Machines), and, BP-MLL neural networks.

Furthermore, 70% were utilized for training out of the entire dataset used for experimentation with these methods, and 30% were used for testing. Each test data set was labeled and the computation of metrics such as hamming-loss, accuracy and log-loss was performed for each method using forecasts and labels. The final findings have been achieved based on combined hamming-loss and log-loss values acquired using computational models.

3.**Evaluation Metrics**

→ The measurements of label bases include one mistake, mean accuracy, etc. The labels may be computed for each label and then measured for all without any relationship between the labels being taken into consideration.

Average accuracy: Average accuracy is a metric that combines call-back and accuracy with the ranking of the results. For one information needs, average accuracy is the means of accuracy ratings, where and at threshold precision is found after each corresponding document is obtained.

→ **Example based metrics** Include precision, lack of hamming, etc. These are computed and averaged for each of the instances. Let –

Accuracy is defined as the proportion of properly forecast labels per occurrence to the total no of labels.

Hamming loss is defined as the symmetrical difference in the division of the total no. of labels between predicted and real labels.

Looking at the data, we see that every 1 out of 10 samples is toxically harmful, every 1 out of 50 samples is obscene and offensive, but the events are quite uncommon. We therefore have skewed information and metric accuracy doesn't provide us the outcomes needed. We shall thus use hamming loss as an assessment parameter.

2. Analysis

1.**Data Exploration**

The following fields comprise of the -

- id: An 8-digit integer value for the individual who made this remark to determine the identify

- comment_text: A multi-line text field with the remark not filtered
- toxic: Binary tag containing 0/1 (0 for no and 1 for yes)
- severe_toxic: Binary tag containing 0/1
- obscene: Binary label that includes 0/1
- threat: Binary label containing 0/1
- insult: binary label containing 0/1
- identity_hate: Binary tag containing 0/1

The comment_text field is preprocessed from these fields and fitted into several classifiers to determine if it is one or more labels/outcome variables (i.e. toxic, severe_toxic, obscene, threat, insult and identity_hate).

```
In [2]: training=pd.read_csv('train.csv')
        training.head()
```

Out[2]:

	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

```
In [3]: testing=pd.read_csv('test.csv')
        testing.head()
```

Out[3]:

	id	comment_text
0	00001cee341fdb12	Yo bitch Ja Rule is more succesful then you'll...
1	0000247867823ef7	== From RfC == \n\n The title is fine as it is...
2	00013b17ad220c46	" \n\n == Sources == \n\n * Zawe Ashton on Lap...
3	00017563c3f7919a	:If you have a look back at the source, the in...
4	00017695ad8997eb	I don't anonymously edit articles at all.

"read_csv" is an important function of pandas which allows to read csv files and we can make various operations on the dataset. As my file is a CSV file that's why I have used "read_csv" function to load the data from the specific directory.The name of the dataset id testing and training.

```
In [4]: print('training shape is ',training.shape)
        print('testing shape is ',testing.shape)
        print('testing info',testing.info)


        print('training info',training.info)
```

```
training shape is  (159571, 8)
testing shape is  (153164, 2)
testing info <bound method DataFrame.info of                      id                                       comment_text
0       00001cee341fdb12  Yo bitch Ja Rule is more succesful then you'll...
1       0000247867823ef7  == From RfC == \n\n The title is fine as it is...
2       00013b17ad220c46  " \n\n == Sources == \n\n * Zawe Ashton on Lap...
3       00017563c3f7919a  :If you have a look back at the source, the in...
4       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]>
training info <bound method DataFrame.info of                      id                                       comment_text  \
0       0000997932d777bf  Explanation\nWhy the edits made under my usern...
1       000103f0d9cfb60f  D'aww! He matches this background colour I'm s...
2       000113f07ec002fd  Hey man, I'm really not trying to edit war. It...
3       0001b41b1c6bb37e  "\nMore\nI can't make any real suggestions on ...
...                  ...                                                ...
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]>
```

Normaly to explore the data we can use various functions such as shape, columns, dtypes, info(), head(), tail(), describe(). Here, I have used training.info(),testing.info(),training.shape(),testing.shape()
By using info() we can get a concise summary of a DataFrame. It includes the index dtype and column dtypes, non-null values and memory usage.
In our dataset we can see that there are 8 columns and `159571 rows for training dataset. And for testing dataset there are two columns and 153164 rows.`

```
In [5]: print('training data Set descriptin',training.describe())
        print('testing data Set descriptin',testing.describe())
```

```
training data Set descriptin           malignant  highly_malignant           rude         threat  \
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
testing data Set descriptin                      id                                         comment_text
count             153164                                                  153164
unique            153164                                                  153164
top     ae063834fb987a10  Hinduism is monotheistic when taken to its log...
freq                   1                                                       1
```

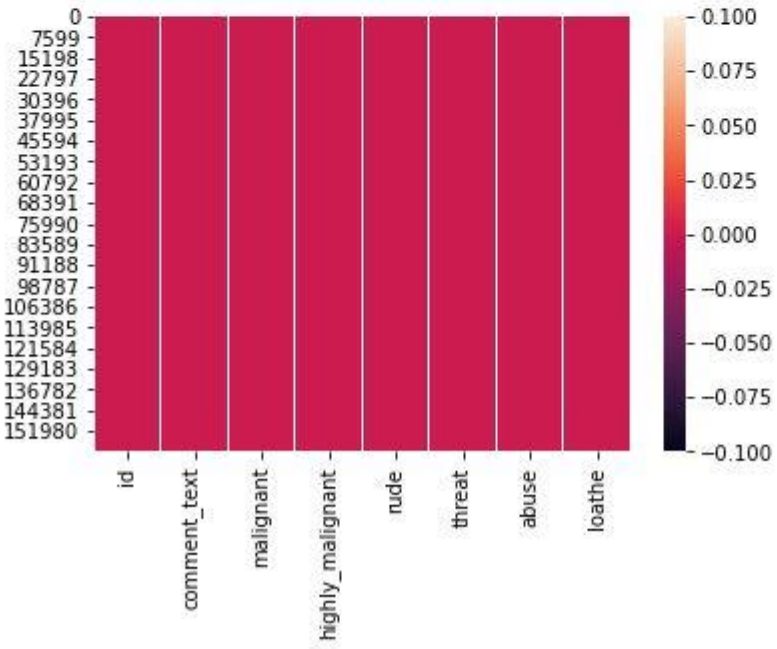
By using describe() function we can explore the count, mean , median, standard deviation, minimum value, 25th, 50th and 75th percentile , maximum value.

2.**Exploratory Visualisation**

1. Checking Null values :

```
In [6]: # checking null values
        print(training.isnull().sum())
        print(sns.heatmap(training.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)
```



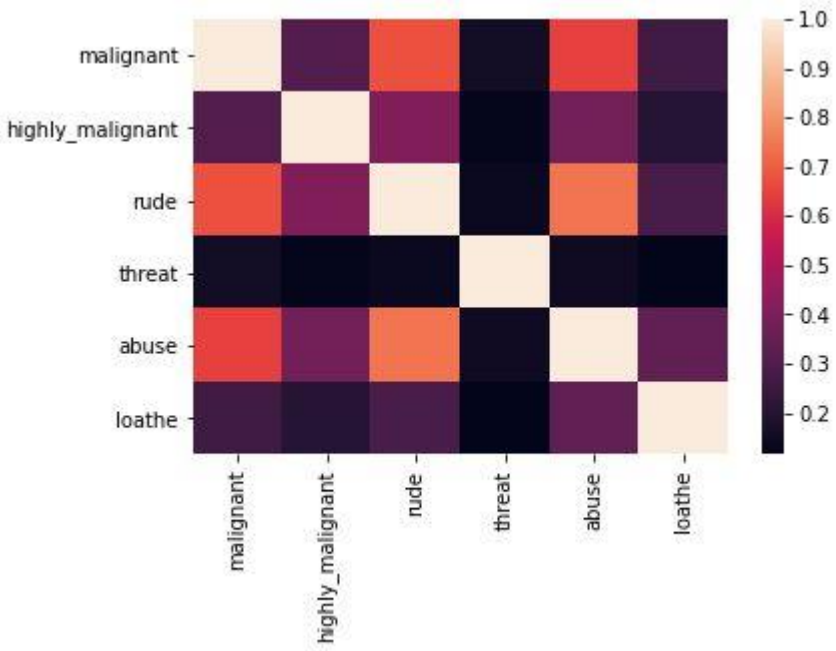
In any colulm there are no Null values in the datset.
If we want to visualizise the null values then there are another method called heatmap. Here, we can see that in nocolumn there are Null Values.

2. Checking Correlation :

```
In [7]: ## checking correlation in dataset
        print(training.corr())
        print(sns.heatmap(training.corr()))
```

```
                  malignant  highly_malignant      rude    threat     abuse  \
malignant          1.000000          0.308619  0.676515  0.157058  0.647518
highly_malignant   0.308619          1.000000  0.403014  0.123601  0.375807
rude               0.676515          0.403014  1.000000  0.141179  0.741272
threat             0.157058          0.123601  0.141179  1.000000  0.150022
abuse              0.647518          0.375807  0.741272  0.150022  1.000000
loathe             0.266009          0.201600  0.286867  0.115128  0.337736

                   loathe
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)
```



From the above figure we we be able to know the exact correlation of each column with the target variable.
The findings are mentioned below:

1. Negative correlation : There are no negative correlation.

2. Positive correlation : All the correlations are positive

3. Strong correlation : (`malignant, abuse),(rude,malignant),(rude,abuse)`

3. Methodology

1.Data Preprocessing

To process the data, the following procedures have been taken:

→ **A string without all punctuations to be prepared:**

1. There are punctuation characters in the string library. All numbers will be added on that line. This is imported. Our field of comment text contains strings that contain apostrophe character(') like will not, did not, etc. The letter 'expressed as \' is substituted in the escape sequence notation with empty character within the punctuation string to avoid converting these words to wont or didn't.
2. make_trans(intab, outtab) function is used. It returns a translation table that maps each character in the intab into the character at the same position in the outtab string.

→ **Updating the list of stop words:**

```
print (stop_words)
```

```
['a', 'about', 'above', 'after', 'again', 'against', 'all', 'am', 'an', 'and', 'any', 'are', "aren't", 'as', 'at', 'be',
'because', 'been', 'before', 'being', 'below', 'between', 'both', 'but', 'by', "can't", 'cannot', 'could', "couldn't", 'di
d', "didn't", 'do', 'does', "doesn't", 'doing', "don't", 'down', 'during', 'each', 'few', 'for', 'from', 'further', 'had',
"hadn't", 'has', "hasn't", 'have', "haven't", 'having', 'he', "he'd", "he'll", "he's", 'her', 'here', "here's", 'hers', 'h
erself', 'him', 'himself', 'his', 'how', "how's", 'i', "i'd", "i'll", "i'm", "i've", 'if', 'in', 'into', 'is', "isn't", 'i
t', "it's", 'its', 'itself', "let's", 'me', 'more', 'most', "mustn't", 'my', 'myself', 'no', 'nor', 'not', 'of', 'off', 'o
n', 'once', 'only', 'or', 'other', 'ought', 'our', 'ours', 'ourselves', 'out', 'over', 'own', 'same', "shan't", 'she', "sh
e'd", "she'll", "she's", 'should', "shouldn't", 'so', 'some', 'such', 'than', 'that', "that's", 'the', 'their', 'theirs',
'them', 'themselves', 'then', 'there', "there's", 'these', 'they', "they'd", "they'll", "they're", "they've", 'this', 'tho
se', 'through', 'to', 'too', 'under', 'until', 'up', 'very', 'was', "wasn't", 'we', "we'd", "we'll", "we're", "we've", 'we
re', "weren't", 'what', "what's", 'when', "when's", 'where', "where's", 'which', 'while', 'who', "who's", 'whom', 'why',
"why's", 'with', "won't", 'would', "wouldn't", 'you', "you'd", "you'll", "you're", "you've", 'your', 'yours', 'yourself',
'yourselves', '', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v',
'w', 'x', 'y', 'z']
```

1. Terms stop are words that are often used both in written and in spoken communication and thus do not have a good or negative influence on our declaration such as "is, this, us and so forth."
2. Single letter words, either existent or formed, do not communicate a meaningful meaning and can be deleted directly. Letters b to z are therefore added to the immediately imported stop words list.

→ **Stemming and Lemmatizing:**

The procedure is called stemming to turn inflected/derived words into the word tree. Many comparable terms of origin are changed into the same word, such as "systems," "stembers," "stembing" etc.

1. Lemmatizing is the technique through which the inflected forms of a word may be grouped into one item. It is very much comparable to the original work, but differs because the intended component of a word in a phrase and in the broader context surrounding that sentence such as adjacent sentences or even a complete document are accurately identified.

2. For this reason, the wordnet library in nltk is utilized. Stemmer and Lemmatizer from nltk are also being imported.

→ **Applying Count Vectorizer:**

1. A string of words can be converted into a text matrix with column headings, represented by words and the values that mean the frequency of the Count Vectorizer word appearance.
2. Stop words have been accepted, converted to reduced and their parameters regularly expressed. Here, we provide our own list of previously established stopwords and options. The default value of the regular expression is.

→ **Converting, replacing and cleaning:**

```
training['comment_text'] = training['comment_text'].str.replace(r'\d+(\.\d+)?', 'numbr')


training['comment_text'] = training['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'])
training['comment_text'] = training['comment_text'].apply(lambda x: ' '.join(
    term for term in x.split() if term not in stop_words))

lem=WordNetLemmatizer()
training['comment_text'] = training['comment_text'].apply(lambda x: ' '.join(
 lem.lemmatize(t) for t in x.split()))
```

1. Convert all messages to lower case
2. Replace email addresses with 'email'
3. Replace URLs with 'webaddress'
4. Replace money symbols with 'moneysymb' (£ can by typed with ALT key + 156)
5. Replace 10 digit phone numbers (formats include paranthesis, spaces, no spaces, dashes) with 'phonenumber'
6. Replace numbers with 'numbr'


```
training['comment_text'] = training['comment_text'].str.replace(r'\d+(\.\d+)?', 'numbr')


training['comment_text'] = training['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'])
training['comment_text'] = training['comment_text'].apply(lambda x: ' '.join(
    term for term in x.split() if term not in stop_words))

lem=WordNetLemmatizer()
training['comment_text'] = training['comment_text'].apply(lambda x: ' '.join(
 lem.lemmatize(t) for t in x.split()))
```


→ **Removing length:**

```
training['clean_length'] = training.comment_text.str.len()
training.head()
```

	id	comment_text	malignant	highly_malignant	rude	threat	abuse	loathe	length	clean_length
0	0000997932d777bf	explanation edits made username hardcore metal...	0	0	0	0	0	0	264	180
1	000103f0d9cfb60f	d'aww! match background colour i'm seemingly s...	0	0	0	0	0	0	112	111
2	000113f07ec002fd	hey man, i'm really trying edit war. guy const...	0	0	0	0	0	0	233	149
3	0001b41b1c6bb37e	can't make real suggestion improvement wondere...	0	0	0	0	0	0	622	397
4	0001d958c54c6e35	you, sir, hero. chance remember page that's on?	0	0	0	0	0	0	67	47

```
# Total Length removal
print ('Origian Length', training.length.sum())
print ('Clean Length', training.clean_length.sum())
```

```
Origian Length 62893130
Clean Length 43575187
```

We can see that the Origian Length was 62893130 . But after cleaning the new Length is 43575187

→Getting sense of loud words :

```
#Getting sense of loud words which are offensive
import wordcloud
from wordcloud import WordCloud
hams = training['comment_text'][training['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()
```

2.Implementation

→ Importing all the model libraries :

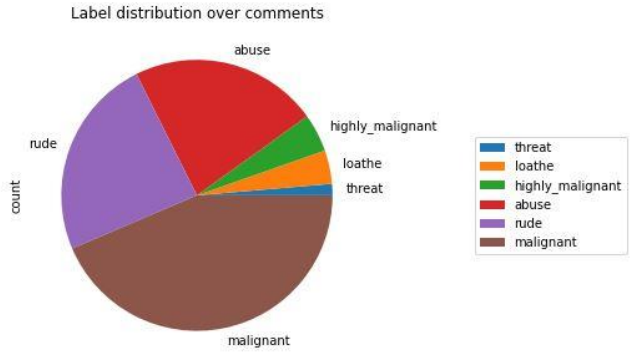
In [17]:
```
from sklearn.naive_bayes import MultinomialNB
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report,roc_curve,roc_auc_score,auc
from sklearn.model_selection import train_test_split
from sklearn.metrics import 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
```

→ Labeled the distribution over comment:

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

df_distribution.plot.pie(y='count',
                                    title='Label distribution over comments',
                                    figsize=(5, 5))\
                            .legend(loc='center left', bbox_to_anchor=(1.3, 0.5))
```

:
```
<matplotlib.legend.Legend at 0x20d7f910b20>
```

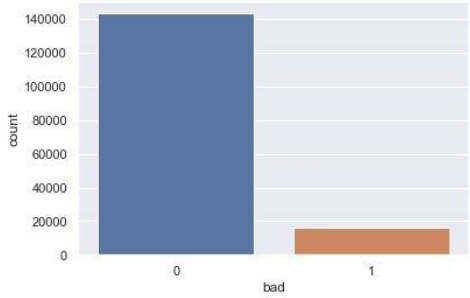


In [19]:
```
target_data = training[cols_target]

training['bad'] =training[cols_target].sum(axis =1)
print(training['bad'].value_counts())
training['bad'] = training['bad'] > 0
training['bad'] = training['bad'].astype(int)
print(training['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
```

In [20]:
```
sns.set()
sns.countplot(x="bad" , data = training)
plt.show()
```



→ Convert the text into TF-IDF vector :

→ Splitting the data into training and testing set :

In [21]:
```
#  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(training['comment_text'])
x = features
```

In [22]:
```
training.shape
```

Out[22]: (159571, 11)

In [23]:
```
testing.shape
```

Out[23]: (153164, 2)

In [24]:
```
y=training['bad']
x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=56,test_size=.30)
```

In [25]:
```
y_train.shape,y_test.shape
```

Out[25]: ((111699,), (47872,))

4. Results

→ Using different algorithms to check the accuracy :

```
# 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 0.9595609629450577
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 0.9395471256684492
[[41599  1351]
 [ 1543  3379]]
              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 0.9554227941176471
[[42421   529]
 [ 1605  3317]]
              precision    recall  f1-score   support

           0       0.96      0.99      0.98     42950
           1       0.86      0.67      0.76      4922

    accuracy                           0.96     47872
   macro avg       0.91      0.83      0.87     47872
weighted avg       0.95      0.96      0.95     47872
```

```
# xgboost
!pip install 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))
```

```
Collecting xgboost
  Downloading xgboost-1.4.2-py3-none-win_amd64.whl (97.8 MB)
Requirement already satisfied: scipy in c:\users\rashi\anaconda3\lib\site-packages (from xgboost) (1.5.2)
Requirement already satisfied: numpy in c:\users\rashi\anaconda3\lib\site-packages (from xgboost) (1.19.2)
Installing collected packages: xgboost
Successfully installed xgboost-1.4.2
```

```
C:\Users\rashi\Anaconda3\lib\site-packages\xgboost\sklearn.py:1146: UserWarning: The use of label encoder i
eprecated and will be removed in a future release. To remove this warning, do the following: 1) Pass option
alse when constructing XGBClassifier object; and 2) Encode your labels (y) as integers starting with 0, i.e
_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)
```

```
[14:34:41] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.4.0/src/learner.cc:1095: Start
0, the default evaluation metric used with the objective 'binary:logistic' was changed from 'error' to 'log
t eval_metric if you'd like to restore the old behavior.
Training accuracy is 0.9614052050600274
Test accuracy is 0.9526236631016043
[[42689   261]
 [ 2007  2915]]
```

```
#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 0.9548587901069518
```

By using all the algoriths one by one we can use one function to implement all the algorithms. If we summarize the result we get the following Accuracy :

1. Accuracy of DecisionTreeClassifier() is :

Training accuracy is 0.9988898736783678

Test accuracy is 0.9395471256684492

2. Accuracy of KNeighborsClassifier() is :

Training accuracy is 0.922300110117369

Test accuracy is 0.9173629679144385

3. Accuracy of RandomForestClassifier() is :

Training accuracy is 0.9988809210467416

Test accuracy is 0.9554227941176471

4. Accuracy of xgboostClassifier() is :

Training accuracy is 0.9614052050600274

Test accuracy is 0.9526236631016043

5. Accuracy of AdaBoostClassifier() is :

Training accuracy is 0.951118631321677

Test accuracy is 0.9490307486631016

6. Accuracy of LogisticClassifier() is :

Training accuracy is 0.9595609629450577

Test accuracy is 0.9552974598930482

7. Accuracy of RandomForestClassifier() is :

Training accuracy is 0.9988898736783678

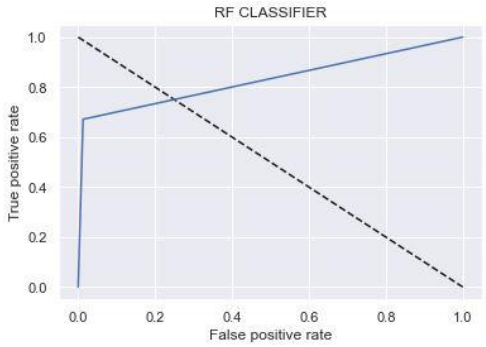
Test accuracy is 0.9548587901069518

We have got good Accuracy by using the following algorithms :

RandomForestClassifier()

But out of these algorithms RandomForestClassifier() is giving the best result. We will use this model.

```
#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()
```



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

```
!pip install eli5
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
```

Weight	Feature
0.0756 ± 0.0596	fuck
0.0403 ± 0.0472	fucking
0.0295 ± 0.0330	shit
0.0209 ± 0.0162	suck
0.0198 ± 0.0119	idiot
0.0184 ± 0.0133	stupid
0.0166 ± 0.0184	bitch
0.0161 ± 0.0141	asshole
0.0116 ± 0.0121	dick
0.0106 ± 0.0058	gay
0.0102 ± 0.0092	faggot
0.0100 ± 0.0102	cunt
0.0083 ± 0.0066	hell
0.0068 ± 0.0075	ass
0.0066 ± 0.0049	bullshit
… 9985 more …	

```
test_data =tf_vec.fit_transform(testing['comment_text'])
test_data
```

```
<153164x10000 sparse matrix of type '<class 'numpy.float64'>'
        with 2940344 stored elements in Compressed Sparse Row format>
```

```
prediction=RF.predict(test_data)
prediction
```

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

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

```
['malig.pkl']
```

Finally, I have saved the file as pkl file

5. Conclusion

2.Reflection

- → This is the summary of the projects we have carried out :
  - → The initial stage was to collect data and decide which part of it was appropriate for training: This step was extremely important because only very small commentaries would produce poor results if the length was increased, while very long commentaries would dramatically increase the number of words, thus increasing training time and causing the system (jupyter kernel) to leave the memory and eventually die. This step would be extremely important.
- → The second major step was performing cleaning of data including punctuation removal, stop word removal, stemming and lemmatizing: This step was also crucial since the occurrence of similar origin words but having different spellings will intend to give similar classification, but computer cannot recognize this on its own. Hence, this step helped to a large extent in both removing and modifying existing words.
- → The third phase consisted of picking the following models: Since I had a wide range of models and classifiers (not restrictive) along with a number of adaptive models in BP-MLL, I choose which models I had a lot to work on for all training and testing.

→ Finally, on the basis of several measurement metrics: I wanted to compare two key assessment metrics: hamming loss and log-loss. Thus, on the combination of these two losses, the final model selection was carried out.

3. Improvement

→ Although we have tested a number of factors in refinement of my model, a better model might exist that offers more precision.
→ Yes. The Adaboost.MH decision tree model we had planned to utilize was not fully implemented by the team. This model is not even mentioned in the Science Sets Multilearn Library. In terms of practical specifics, the Research Papers were also a little unclear.

3.**Future Scope**

The current project forecasts the kind or toxicity of the observation. In the future we aim to include these characteristics :

→ Analysis of the age group that is harmful to a group or brand.
→ Sensitivize words considered as hazardous by adding feature.
→ If threats are deemed as severe, they are automatically sent to the competent authority.
→ Create a feedback loop to enhance the model's effectiveness.
→ Treat errors and short words to make the outcome more accurate.

References

- Wikipedia : https://en.wikipedia.org/wiki/Multi-label_classification
- Kaggle challenge page for datasets and ideas : https://www.kaggle.com/c/
- jigsaw-toxic-comment-classification-challenge
- Conversation AI git page : https://conversationai.github.io/
- Research Paper titled “Multi-label Classification: Problem Transformation methods in Tamil Phoneme classification” : https://ac.els-cdn.com/
- S1877050917319440/1-s2.0-S1877050917319440-main.pdf? _tid=eced1a38-f8fa-11e7-b8ef-00000aab0f27&acdnat=1515914406_0f244d3e6313bb049c435bf43 504bd52
- Research Paper titled “Benchmarking Multi-label Classification Algorithms” :
- http://ceur-ws.org/Vol-1751/AICS_2016_paper_33.pdf
- Problem Transformation Methods : https://www.analyticsvidhya.com/blog/2017/08/introduction-to-multi-label-classification/
- Research Paper on BP-MLL : http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.507.910&rep=rep1&type=pdf
- GridsearchCV on Sequential Models : https://dzubo.github.io/machine-learning/2017/05/25/increasing-model-accuracy-by-tuning-parameters.html

- Ml-knn - A Lazy Learning Approach to Multi-Label Learning : https://cs.nju.edu.cn/zhouzh/zhouzh.files/publication/pr07.pdf
- Average Precision score : http://scikit-learn.org/stable/modules/generated/sklearn.metrics.average_precision_score.html