FLIP ROBO

MALIGNANT COMMENTS CLASSIFIER PROJECT

Submitted by:
Laxmikant Deepak

ACKNOWLEDGMENT

I have referred below resources that helped and guided me in completion of this project as below :-

https://www.indianaiproduction.com

https://www.patreon.com/dataschool

INTRODUCTION

BUSINESS PROBLEM FRAMING

- The proliferation of social media enables people to express their opinions widely online. However, at the same time, this has resulted in the emergence of conflict and hate, making online environments uninviting for users. Although researchers have found that hate is a problem across multiple platforms, there is a lack of models for online hate detection.
- Online hate, described as abusive language, aggression, cyberbullying, hatefulness and many others has been identified as a major threat on online social media platforms. Social media platforms are the most prominent grounds for such toxic 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 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

- In the past few years its seen that the cases related to social media hatred have increased exponentially. The social media is turning into a dark venomous pit for people now a days. Online hate is the result of difference in opinion, race, religion, occupation, nationality etc.

- In social media the people spreading or involved in such kind of activities uses filthy languages, aggression, images etc. to offend and gravely hurt the person on the other side. This is one of the major concerns now.
- The result of such activities can be dangerous. It gives mental trauma to the victims making their lives miserable. People who are not well aware of mental health online hate or cyber bullying become life threatening for them. Such cases are also at rise. It is also taking its toll on religions. Each and every day we can see an incident of fighting between people of different communities or religions due to offensive social media posts.
- Online hate, described as abusive language, aggression, cyberbullying, hatefulness, insults, personal attacks, provocation, racism, sexism, threats, or toxicity has been identified as a major threat on online social media platforms. These kinds of activities must be checked for a better future.

REVIEW OF LITERATURE

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.

MOTIVATION FOR THE PROBLEM UNDERTAKEN

The project was the first provided to me by FlipRobo as a part of the internship programme. The exposure to real world data and the opportunity to deploy my skillset in solving a real time problem has been the primary objective. However, the motivation for taking this project was that it is relatively a new field of research. Here we have many options but less concrete solutions. The main motivation 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.

.

ANALYTICAL PROBLEM FRAMING

MATHEMATICAL/ ANALYTICAL MODELING OF THE PROBLEM

Here we are dealing with one main text columns which held some importance of the data and others shows the multiple types of behaviour inferred from the text. I prefer to select on focus more on the words which has great value of importance in the context. Countvector is the NLP terms I am going to apply on text columns. This converts the important words proper vectors with some weights.

DATA SOURCES AND THEIR FORMATS

The data was provided by FlipRobo in CSV format. After loading the training dataset into Jupyter Notebook using Pandas and it can be seen that there are eight columns named as:

“id, comment_text, “malignant, highly_malignant, rude, threat, abuse, loathe**”.**

There are 8 columns in the dataset provided:

The description of each of the column is given below:

- **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.

In [8]:
```
# Information of the train dataframe.
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
```

In [10]:
```
# Check the features, duplicate values and nan values in the Datasets

print("\nFeatures Present in the Dataset: \n", df_train.columns)
shape=df_train.shapeS
print("\nTotal Number of Rows : ",shape[0])
print("Total Number of Features : ", shape[1])
print("\n\nData Types of Features :\n", df_train.dtypes)
print("\nDataset contains any NaN/Empty cells : ", df_train.isnull().values.any())
print("\nTotal number of empty rows in each feature:\n", df_train.isnull().sum(),"\n\n")
print("Total number of unique values in each feature:")
for col in df_train.columns.values:
    print("Number of unique values of {} : {}".format(col, df_train[col].nunique()))
```

```
Features Present in the Dataset:
 Index(['id', 'comment_text', 'malignant', 'highly_malignant', 'rude', 'threat',
       'abuse', 'loathe'],
      dtype='object')

Total Number of Rows :  159571
Total Number of Features :  8


Data Types of Features :
 id                  object
comment_text        object
malignant            int64
highly_malignant     int64
rude                 int64
threat               int64
abuse                int64
loathe               int64
dtype: object

Dataset contains any NaN/Empty cells :  False

Total number of empty rows in each feature:
 id                  0
comment_text        0
malignant           0
highly_malignant    0
rude                0
threat              0
abuse               0
loathe              0
dtype: int64


Total number of unique values in each feature:
Number of unique values of id : 159571
Number of unique values of comment_text : 159571
Number of unique values of malignant : 2
Number of unique values of highly_malignant : 2
Number of unique values of rude : 2
Number of unique values of threat : 2
Number of unique values of abuse : 2
Number of unique values of loathe : 2
```

```
In [11]: # Check value counts for each feature

         cols=['malignant', 'highly_malignant', 'rude', 'threat','abuse', 'loathe',]
         for col in cols:
             print("Number of value_counts of {} : {}".format(col, df_train[col].nunique()))
             print(df_train[f'{col}'].value_counts())

         Number of value_counts of malignant : 2
         0    144277
         1     15294
         Name: malignant, dtype: int64
         Number of value_counts of highly_malignant : 2
         0    157976
         1      1595
         Name: highly_malignant, dtype: int64
         Number of value_counts of rude : 2
         0    151122
         1      8449
         Name: rude, dtype: int64
         Number of value_counts of threat : 2
         0    159093
         1       478
         Name: threat, dtype: int64
         Number of value_counts of abuse : 2
         0    151694
         1      7877
         Name: abuse, dtype: int64
         Number of value_counts of loathe : 2
         0    158166
         1      1405
         Name: loathe, dtype: int64
```

DATA PREPROCESSING DONE

After loading all the required libraries we loaded the data into our jupyter notebook.

```
In [1]: # Importing all the required libraries.

        import pandas as pd
        import numpy as np
        import seaborn as sns
        import matplotlib.pyplot as plt
        from collections import Counter
        import string
        import re

        # packages from gensim
        from gensim import corpora
        from gensim.parsing.preprocessing import STOPWORDS
        from gensim.utils import simple_preprocess

        # packages from sklearn
        from sklearn.feature_extraction.text import TfidfVectorizer

        # packages from nltk
        import nltk
        from nltk.corpus import wordnet
        from nltk.stem import WordNetLemmatizer, SnowballStemmer
        from nltk import pos_tag

        import warnings
        warnings.filterwarnings('ignore')
```

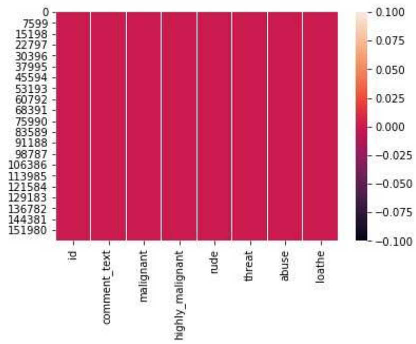
Feature Engineering has been used for cleaning of the data. We first did data cleaning. We first looked percentage of values missing in columns.

```
In [12]: # Finding null values for train dataset.
         df_train.isnull().sum()

Out[12]: id                  0
         comment_text        0
         malignant           0
         highly_malignant    0
         rude                0
         threat              0
         abuse               0
         loathe              0
         dtype: int64
```

Observation:

We do not have any null values in our dataset.

```
In [13]: #checking null values using heatmap
         sns.heatmap(df_train.isnull());
```



Observation:

There are no Null values in this dataset.

For Data pre-processing we did some data cleaning, where we used wordNetlemmatizerto clean the words and removed special characters using Regexp Tokenizer and filter the words by removing stop words and then used lemmatizers and joined and return the filtered words.

Used TFIDF vectorizer to convert those text into vectors, and split the data and into test and train and trained various Machine learning algorithms.

```
In [31]: #Creating a function to filter using POS tagging.

         def get_pos(pos_tag):
             if pos_tag.startswith('J'):
                 return wordnet.ADJ
             elif pos_tag.startswith('N'):
                 return wordnet.NOUN
             elif pos_tag.startswith('R'):
                 return wordnet.ADV
             else:
                 return wordnet.NOUN
```

In [32]:
```
# Function for data cleaning...
def Processed_data(comments):
    # Replace email addresses with 'email'
    comments=re.sub(r'^.+@[^\.].*\.[a-z]{2,}$',' ', comments)

    # Replace 10 digit phone numbers (formats include paranthesis, spaces, no spaces, dashes) with 'phonenumber'
    comments=re.sub(r'^\(?[\d]{3}\)?[\s-]?[\d]{3}[\s-]?[\d]{4}$',' ',comments)

    # getting only words(i.e removing all the special characters)
    comments = re.sub(r'[^\w]', ' ', comments)

    # getting only words(i.e removing all the" _ ")
    comments = re.sub(r'[\_]', ' ', comments)

    # getting rid of unwanted characters(i.e remove all the single characters left)
    comments=re.sub(r'\s+[a-zA-Z]\s+', ' ', comments)

    # Removing extra whitespaces
    comments=re.sub(r'\s+', ' ', comments, flags=re.I)

    #converting all the letters of the review into lowercase
    comments = comments.lower()

    # splitting every words from the sentences
    comments = comments.split()

    # iterating through each words and checking if they are stopwords or not,
    comments=[word for word in comments if not word in set(STOPWORDS)]

    # remove empty tokens
    comments = [text for text in comments if len(text) > 0]

    # getting pos tag text
    pos_tags = pos_tag(comments)

    # considering words having length more than 3only
    comments = [text for text in comments if len(text) > 3]

    # performing lemmatization operation and passing the word in get_pos function to get filtered using POS ...
    comments = [(WordNetLemmatizer().lemmatize(text[0], get_pos(text[1])))for text in pos_tags]

   # considering words having length more than 3 only
    comments = [text for text in comments if len(text) > 3]
    comments = ' '.join(comments)
    return comments
```

In [33]:
```
# Cleaning  and storing the comments in a separate feature.
df_train["clean_comment_text"] = df_train["comment_text"].apply(lambda x: Processed_data(x))
```

In [34]:
```
# Cleaning and storing the comments in a separate feature.
df_test["clean_comment_text"] = df_test["comment_text"].apply(lambda x: Processed_data(x))
```

In [35]:
```
# Adding new feature clean_comment_length to store length of cleaned comments in clean_comment_text characters
df_train['clean_comment_length'] = df_train['clean_comment_text'].apply(lambda x: len(str(x)))
df_train.head()
```

Out[35]:

	comment_text	malignant	highly_malignant	rude	threat	abuse	loathe	comment_length	label	clean_comment_text	clean_comment_length
0	Explanation\nWhy the edits made under my usern...	0	0	0	0	0	0	264	0	explanation edits username hardcore metallica ...	129
1	D'aww! He matches this background colour I'm s...	0	0	0	0	0	0	112	0	match background colour seemingly stuck thanks...	64
2	Hey man, I'm really not trying to edit war. It...	0	0	0	0	0	0	233	0	trying edit constantly removing relevant infor...	112
3	"\nMore\nI can't make any real suggestions on ...	0	0	0	0	0	0	622	0	real suggestion improvement wondered section s...	315
4	You, sir, are my hero. Any chance you remember...	0	0	0	0	0	0	67	0	hero chance remember page	25

In [36]:
```
df_test['clean_comment_length'] = df_test['clean_comment_text'].apply(lambda x: len(str(x)))
df_test.head()
```

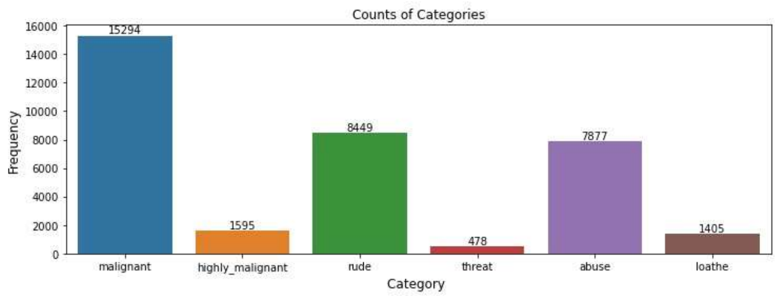
Out[36]:

	id	comment_text	comment_length	clean_comment_text	clean_comment_length
0	00001cee341fdb12	Yo bitch Ja Rule is more succesful then you'll...	367	bitch rule succesful whats hating mofuckas bit...	184
1	0000247867823ef7	== From RfC == \n\n The title is fine as it is...	50	title fine	10
2	00013b17ad220c46	" \n\n == Sources == \n\n * Zawe Ashton on Lap...	54	source zawe ashton lapland	26
3	00017563c3f7919a	:If you have a look back at the source, the in...	205	look source information updated correct form g...	109
4	00017695ad8997eb	I don't anonymously edit articles at all.	41	anonymously edit article	24

DATA INPUTS- LOGIC- OUTPUT RELATIONSHIPS

EDA was performed by creating valuable insights using various visualization libraries.

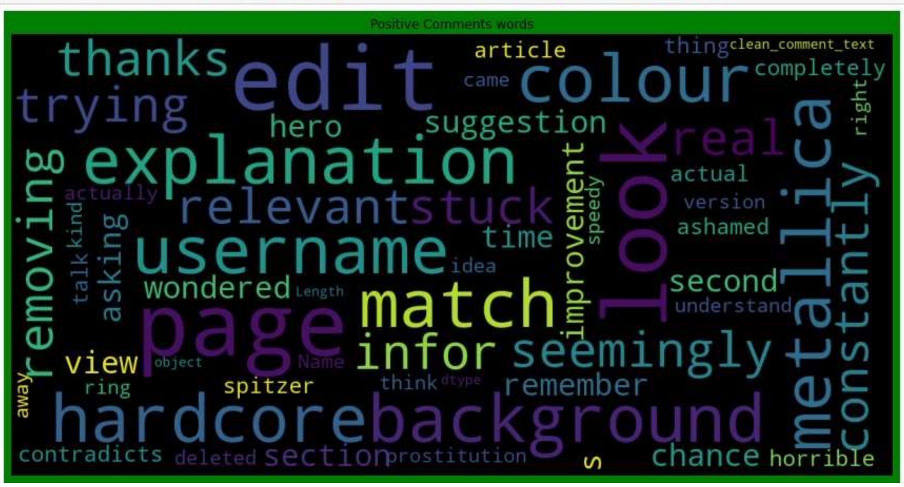
```
In [25]: # Let's plot the counts of each category

plt.figure(figsize=(12,4))
ax = sns.barplot(counts.index, counts.values)
plt.title("Counts of Categories")
plt.ylabel('Frequency', fontsize=12)
plt.xlabel('Category ', fontsize=12)
rects = ax.patches
labels = counts.values
for rect, label in zip(rects, labels):
    height = rect.get_height()
    ax.text(rect.get_x() + rect.get_width()/2, height + 5, label, ha='center', va='bottom')
plt.show()
```




```
In [26]: #Checking percentage of good and bad comments in dataset
good_comments = df_train[(df_train['malignant']!=1) & (df_train['highly_malignant']!=1) & (df_train['rude']!=1) &
                         (df_train['threat']!=1) & (df_train['abuse']!=1) & (df_train['loathe']!=1)]
good_percent=len(good_comments)/len(df_train)*100
print('Percentage of good comments = ',good_percent)
print('Percentage of negative comments = ', (100-good_percent))

Percentage of good comments =  89.83211235124176
Percentage of negative comments =  10.167887648758239
```

Malignant Words:

```
In [38]: # Non-Negative/Good Comments - in training data
Display_wordcloud(df_train['clean_comment_text'][df_train['label']==0],"Positive Comments")
```



NoN Malignant Words:

```
In [39]: # Negative Comments - in training data
         Display_wordcloud(df_train['clean_comment_text'][df_train['label']==1],"Negative Comments")
```

HARDWARE AND SOFTWARE REQUIREMENTS AND TOOLS USED

HARDWARE:

Device specifications Copy

Device name	LAPTOP-N0SDNE9F
Processor	AMD Ryzen 5 4600H with Radeon Graphics 3.00 GHz
Installed RAM	8.00 GB (7.37 GB usable)
Device ID	FD942C3B-FA5D-4994-AD8C-62A017AA85FA
Product ID	00331-10000-00001-AA038
System type	64-bit operating system, x64-based processor
Pen and touch	No pen or touch input is available for this display

Windows specifications Copy

Edition	Windows 11 Pro Insider Preview
Version	Dev
Installed on	04-09-2021
OS build	22449.1000
Serial number	PF20GFKB
Experience	Windows Feature Experience Pack 1000.22449.1000.0

Microsoft Services Agreement
Microsoft Software Licence Terms

SOFTWARE:

Jupyter Notebook (Anaconda 3) – Python 3.8.5

Microsoft Excel 2019

LIBRARIES:

The tools, libraries and packages we used for accomplishing this project are pandas, numpy, matplotlib, seaborn, scipy stats, etc.

```
In [1]: # Importing all the required libraries.

        import pandas as pd
        import numpy as np
        import seaborn as sns
        import matplotlib.pyplot as plt
        from collections import Counter
        import string
        import re

        # packages from gensim
        from gensim import corpora
        from gensim.parsing.preprocessing import STOPWORDS
        from gensim.utils import simple_preprocess

        # packages from sklearn
        from sklearn.feature_extraction.text import TfidfVectorizer

        # packages from nltk
        import nltk
        from nltk.corpus import wordnet
        from nltk.stem import WordNetLemmatizer, SnowballStemmer
        from nltk import pos_tag

        import warnings
        warnings.filterwarnings('ignore')
```

MODEL/S DEVELOPMENT AND EVALUATION

IDENTIFICATION OF POSSIBLE PROBLEM-SOLVING APPROACHES (METHODS)

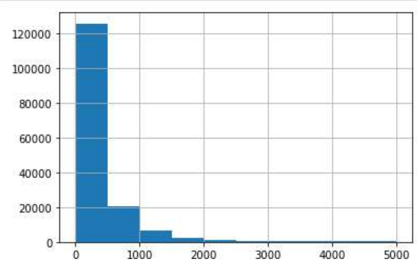
The dataset is loaded and stored in a data frame. We need to perform some text processing to remove unwanted words and characters from our text. I used the nltk library and the string library. Then the data was analysed and visualized to extract insights about the comments. The sentence in the cleaned data, were broken down into vectors using Tokenizer from Keras and each word was converted into sequence of integers. Comments are variable in length, some are one-word replies while others are vastly elaborated thoughts. To overcome this issue, we use Padding. With the help of padding, we can make the shorter sentences as long as the others by filling the shortfall by zeros, and on the other hand, we can trim the longer ones to the same length as the short ones [3]. I used the “pad_sequences” function from the “Keras” library and, I fixed the sentence length at 200 words and applied pre padding (i.e. for shorter sentences, 0’s will be added at the beginning of the sequence vector) A model was built using Keras and Tensorflow. For our classification task, I used both CNN and LSTM neural networks. The model consisted of Embedding layer, which is responsible for embedding. MaxPool layer used to focus on the important features. Bi-directional LSTM was used for one forward and one backward network. Last layer consisted of Sigmoid layer, which will predict probabilities for each kind of features in our dataset. The training dataset was split into training and validation set. 20% of the training data was kept aside for validation. The model was compiled with various optimizers, amongst which adam performed better and metrics like loss and AUC were used to evaluate the model. The dataset was then fit on training data and validated on validation dataset. It gave a quite good AUC of about 98.3% with 2 epochs. The loss was also decreasing significantly with increase in epoch, and finally the model was used to predict on the testing dataset.

TESTING OF IDENTIFIED APPROACHES (ALGORITHMS)

In [47]:
```
# Creating instances for different Classifiers

LR=LogisticRegression()
MNB=MultinomialNB()
DT=DecisionTreeClassifier()
KNN=KNeighborsClassifier()
RFC=RandomForestClassifier()
GBC=GradientBoostingClassifier()
SV=SVC()
```

VISUALIZATIONS

In [19]:
```
# Let's Plot the Length in a histogram

lens.hist();
```



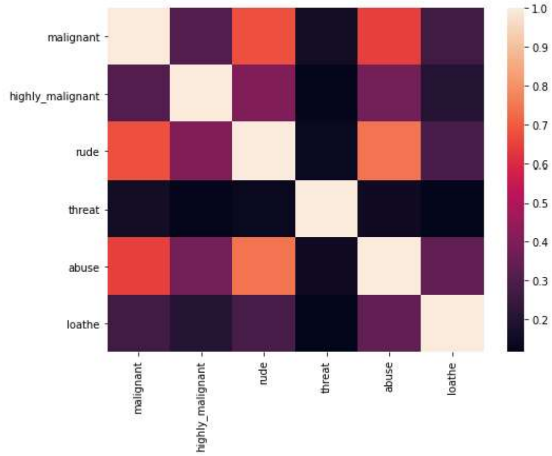
In [20]:
```
# Let's plot the correlation chart

df_train.corr().style.background_gradient(cmap='YlGnBu')
```

Out[20]:

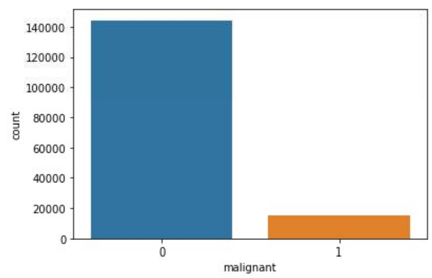
	malignant	highly_malignant	rude	threat	abuse	loathe
malignant	1.000000	0.308619	0.676515	0.157058	0.647518	0.266009
highly_malignant	0.308619	1.000000	0.403014	0.123601	0.375807	0.201600
rude	0.676515	0.403014	1.000000	0.141179	0.741272	0.286867
threat	0.157058	0.123601	0.141179	1.000000	0.150022	0.115128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

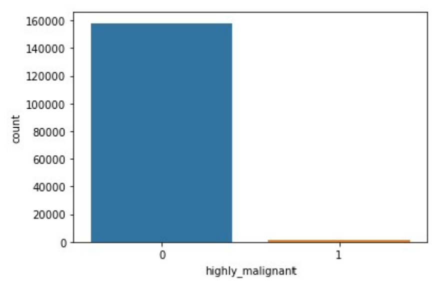
```
In [21]: # Let's view the Correlation heatmap among variables

         plt.figure(figsize=(8,6))
         sns.heatmap(df_train.corr())
```

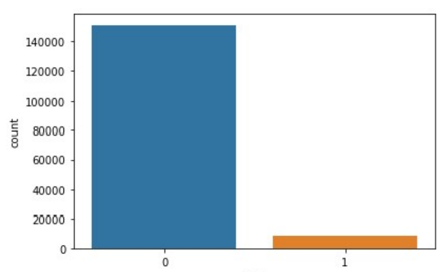
Out[21]: <AxesSubplot:>

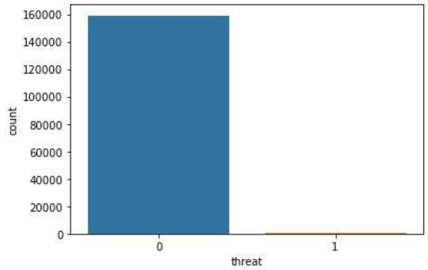


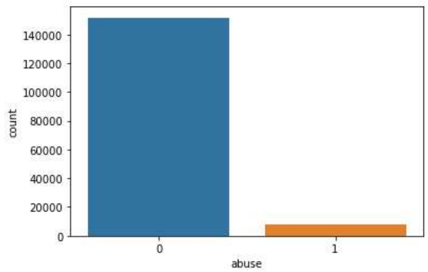

```
In [22]: for i in features:
             sns.countplot(df_train[i])
             plt.show()
```

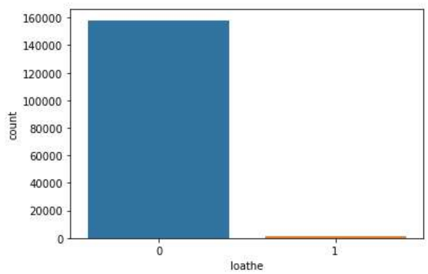






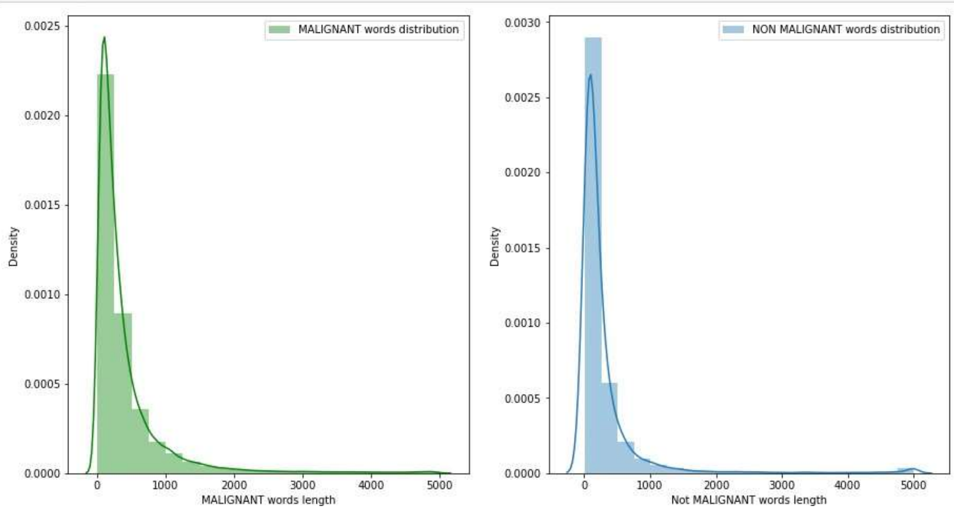




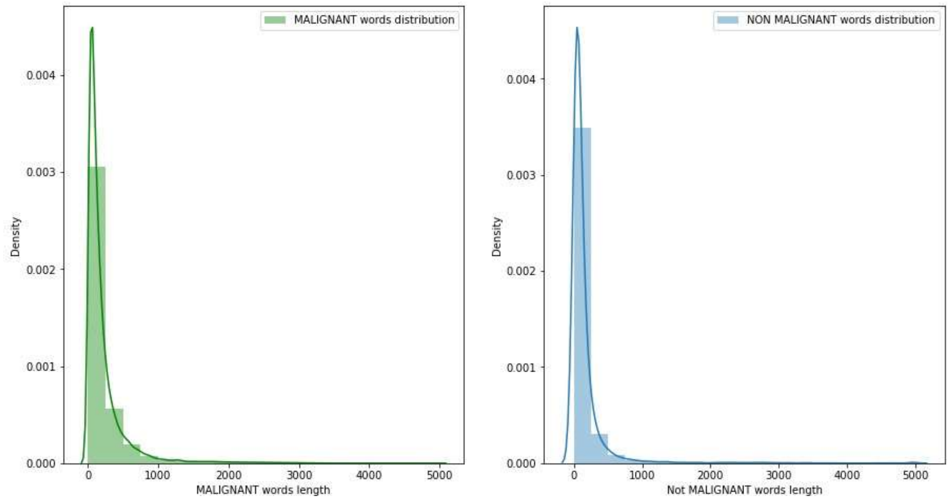


Most of the comments are non-negative but still there are some highly malignant, rude and abuse comments.

In [40]:
```
# Comments length distribution BEFORE cleaning
f,ax = plt.subplots(1,2,figsize = (15,8))

sns.distplot(df_train[df_train['label']==0]['comment_length'],bins=20,ax=ax[0],label='MALIGNANT words distribution',color='g')

ax[0].set_xlabel('MALIGNANT words length')
ax[0].legend()

sns.distplot(df_train[df_train['label']==1]['comment_length'],bins=20,ax=ax[1],label='NON MALIGNANT words distribution')
ax[1].set_xlabel('Not MALIGNANT words length')
ax[1].legend()

plt.show()
```



In [41]:
```
# Comments length distribution after cleaning
f,ax = plt.subplots(1,2,figsize = (15,8))

sns.distplot(df_train[df_train['label']==0]['clean_comment_length'],bins=20,ax=ax[0],label='MALIGNANT words distribution',color='

ax[0].set_xlabel('MALIGNANT words length')
ax[0].legend()

sns.distplot(df_train[df_train['label']==1]['clean_comment_length'],bins=20,ax=ax[1],label='NON MALIGNANT words distribution')
ax[1].set_xlabel('Not MALIGNANT words length')
ax[1].legend()

plt.show()
```



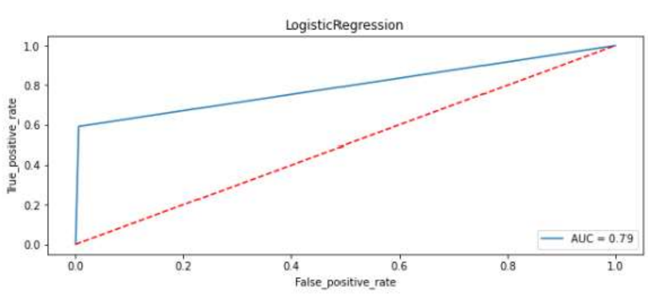
RUN AND EVALUATED SELECTED MODELS

In [47]:
```
# Creating instances for different Classifiers

LR=LogisticRegression()
MNB=MultinomialNB()
DT=DecisionTreeClassifier()
KNN=KNeighborsClassifier()
RFC=RandomForestClassifier()
GBC=GradientBoostingClassifier()
SV=SVC()
```

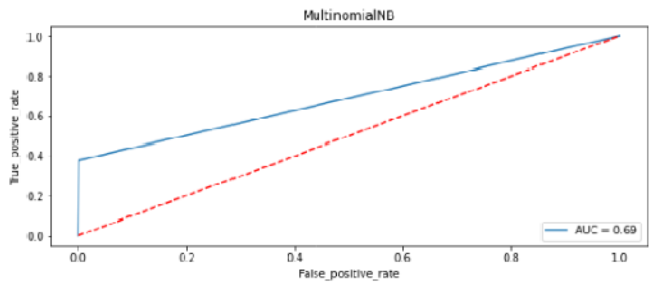
In [48]:
```
# Creating a list model where all the models will be appended for further evaluation in loop.
models=[]
models.append(('LogisticRegression',LR))
models.append(('MultinomialNB',MNB))
models.append(('DecisionTreeClassifier',DT))
models.append(('KNeighborsClassifier',KNN))
models.append(('RandomForestClassifier',RFC))
models.append(('GradientBoostingClassifier',GBC))
models.append(('SVC',SV))
```

In [49]:
```python
# Lists to store model name, Learning score, Accuracy score, cross_val_score, Auc Roc score.

Model=[]
Score=[]
Acc_score=[]
cvs=[]
rocscore=[]
lg_loss=[]

# For Loop to Calculate Accuracy Score, Cross Val Score, Classification Report, Confusion Matrix

for name,model in models:
    print(name)
    Model.append(name)
    print(model)

    x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.30,random_state=42,stratify=y)
    model.fit(x_train,y_train)

# Learning Score
    score=model.score(x_train,y_train)
    print('Learning Score : ',score)
    Score.append(score*100)
    y_pred=model.predict(x_test)
    acc_score=accuracy_score(y_test,y_pred)
    print('Accuracy Score : ',acc_score)
    Acc_score.append(acc_score*100)

# Cross_val_score
    cv_score=cross_val_score(model,x,y,cv=5,scoring='roc_auc').mean()
    print('Cross Val Score : ', cv_score)
    cvs.append(cv_score*100)

# Roc auc score
    false_positive_rate,true_positive_rate, thresholds=roc_curve(y_test,y_pred)
    roc_auc=auc(false_positive_rate, true_positive_rate)
    print('roc auc score : ', roc_auc)
    rocscore.append(roc_auc*100)

# Log Loss
    loss = log_loss(y_test,y_pred)
    print('Log loss : ', loss)
    lg_loss.append(loss)

# Classification Report
    print('Classification Report:\n',classification_report(y_test,y_pred))
    print('\n')

    print('Confusion Matrix:\n',confusion_matrix(y_test,y_pred))
    print('\n')



    plt.figure(figsize=(10,40))
    plt.subplot(911)
    plt.title(name)
    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')
```

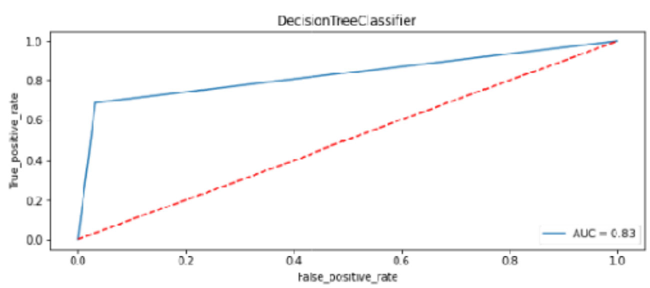
```
LogisticRegression
LogisticRegression()
Learning Score :  0.9577704366198444
Accuracy Score :  0.9531876671122995
Cross Val Score :  0.9640643421763972
roc auc score :  0.7925034414256777
Log loss :  1.616844857134755
Classification Report:
               precision    recall  f1-score   support

           0       0.96      0.99      0.97     43004
           1       0.92      0.59      0.72      4868

    accuracy                           0.95     47872
   macro avg       0.94      0.79      0.85     47872
weighted avg       0.95      0.95      0.95     47872



Confusion Matrix:
 [[42755   249]
 [ 1992  2876]]
```



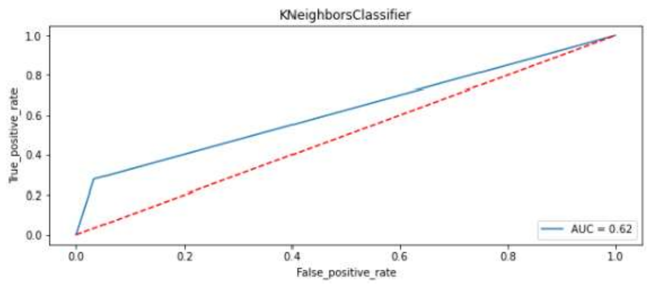
```
MultinomialNB
MultinomialNB()
Learning Score :  0.9397487891565726
Accuracy Score :  0.9354737633589839
Cross Val Score :  0.9264906705491673
roc auc score :  0.6884622511658735
Log loss :  2.2286583108814б
Classification Report:
              precision    recall  f1-score   support

           0       0.93      1.00      0.97     43004
           1       0.97      0.38      0.54      4868

    accuracy                           0.94     47872
   macro avg       0.95      0.69      0.75     47872
weighted avg       0.94      0.94      0.92     47872


Confusion Matrix:
 [[42941    63]
 [ 3026  1842]]
```



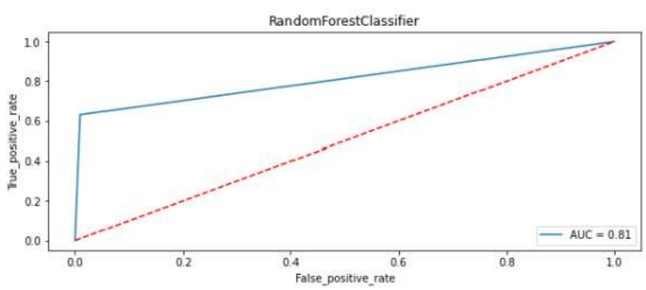

```
DecisionTreeClassifier
DecisionTreeClassifier()
Learning Score :  0.9982631894645431
Accuracy Score :  0.9392337901069518
Cross Val Score :  0.8342348225831232
roc auc score :  0.8270911356624486
Log loss :  2.0988136191б0339
Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.97      0.97     43004
           1       0.71      0.69      0.70      4868

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


Confusion Matrix:
 [[41622  1382]
 [ 1527  3341]]
```



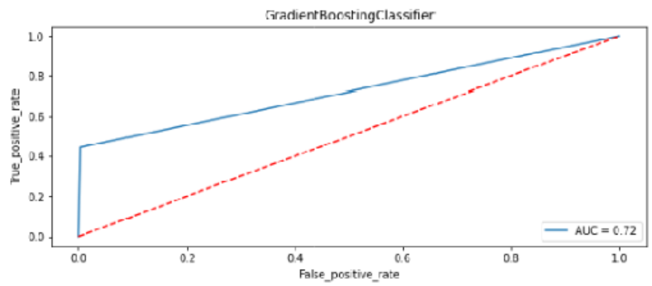
```
KNeighborsClassifier
KNeighborsClassifier()
Learning Score :  0.9235355732817662
Accuracy Score :  0.8970170454545454
Cross Val Score :  0.690548573731317
roc auc score :  0.6223346481995865
Log loss :  3.5569288406182857
Classification Report:
              precision    recall  f1-score   support

           0       0.92      0.97      0.94     43004
           1       0.49      0.28      0.35      4868

    accuracy                           0.90     47872
   macro avg       0.71      0.62      0.65     47872
weighted avg       0.88      0.90      0.88     47872


Confusion Matrix:
 [[41591  1413]
 [ 3517  1351]]
```



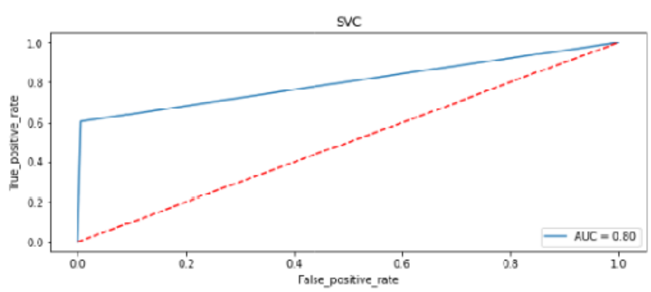

```
RandomForestClassifier
RandomForestClassifier()
Learning Score :  0.9982631894645431
Accuracy Score :  0.9539396724598931
Cross Val Score :  0.9553084714170058
roc auc score :  0.8105924506688224
Log loss :  1.5908741516321059
Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.99      0.97     43004
           1       0.88      0.63      0.74      4868

    accuracy                           0.95     47872
   macro avg       0.92      0.81      0.86     47872
weighted avg       0.95      0.95      0.95     47872


Confusion Matrix:
 [[42597   407]
 [ 1798  3070]]
```



```
GradientBoostingClassifier
GradientBoostingClassifier()
Learning Score :  0.9429985944368348
Accuracy Score :  0.9405015775401069
Cross Val Score :  0.8896284595702081
roc auc score :  0.7214598791024159
Log loss :  2.0518967080369133
Classification Report:
              precision    recall  f1-score   support

           0       0.94      1.00      0.97     43004
           1       0.94      0.45      0.60      4868

    accuracy                           0.94     47872
   macro avg       0.94      0.72      0.79     47872
weighted avg       0.94      0.94      0.93     47872


Confusion Matrix:
 [[42855   149]
 [ 2695  2173]]
```




```
SVC
SVC()
Learning Score :  0.9812352841117646
Accuracy Score :  0.9545872326203209
Cross Val Score :  0.9627966041119127
roc auc score :  0.8002959652833312
Log loss :  1.5685657440313812
Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.99      0.98     43004
           1       0.92      0.61      0.73      4868

    accuracy                           0.95     47872
   macro avg       0.94      0.80      0.85     47872
weighted avg       0.95      0.95      0.95     47872


Confusion Matrix:
 [[42745   259]
 [ 1915  2953]]
```



In [50]:
```
# Displaying scores :
results=pd.DataFrame({'Model': Model,'Learning Score': Score,'Accuracy Score': Acc_score,'Cross Val Score':cvs,
                     'Auc_score':rocscore,'Log_Loss':lg_loss})
results
```

Out[50]:

	Model	Learning Score	Accuracy Score	Cross Val Score	Auc_score	Log_Loss
0	LogisticRegression	95.777044	95.318767	96.406434	79.250344	1.616845
1	MultinomialNB	93.974879	93.547376	92.649067	68.846225	2.228658
2	DecisionTreeClassifier	99.826319	93.923379	83.423482	82.709114	2.098814
3	KNeighborsClassifier	92.353557	89.701705	69.054857	62.233465	3.556929
4	RandomForestClassifier	99.826319	95.393967	95.530847	81.059245	1.590874
5	GradientBoostingClassifier	94.299859	94.059158	88.962846	72.145988	2.051897
6	SVC	98.123528	95.458723	96.279660	80.029597	1.568506

Looking at all the Scores, I have selected Random Forest

Hyperparameter Tuning - Random Forest

In [51]:
```
from sklearn.model_selection import RandomizedSearchCV
x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=42,test_size=.30,stratify=y)
parameters={'bootstrap': [True, False],
 'max_depth': [10, 50, 100, None],
 'min_samples_leaf': [1, 2, 4],
 'min_samples_split': [2, 5, 10],
 'n_estimators': [100, 300, 500, 800, 1200]}


LG=LogisticRegression()

# Applying Randomized Search CV for hyperparameter tuning with scoring= "accuracy"
rand = RandomizedSearchCV(estimator = RFC, param_distributions = parameters,
                          n_iter = 10, cv = 3, verbose=2, random_state=42, n_jobs = -1,scoring='accuracy')
rand.fit(x_train,y_train)
rand.best_params_
```

```
Fitting 3 folds for each of 10 candidates, totalling 30 fits
```

Out[51]:
```
{'n_estimators': 500,
 'min_samples_split': 2,
 'min_samples_leaf': 1,
 'max_depth': 100,
 'bootstrap': False}
```

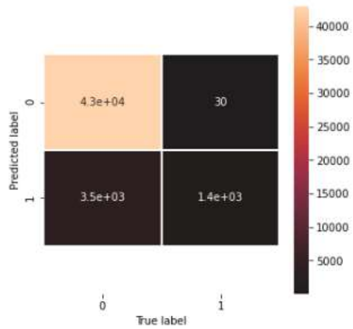
In [52]:
```
RFC=RandomForestClassifier(n_estimators= 500,
                           min_samples_split= 2,
                           min_samples_leaf=1,
                           max_depth= 100,
                           bootstrap= False)
```

In [53]:
```
RFC.fit(x_train,y_train)
RFC.score(x_train,y_train)
pred=RFC.predict(x_test)
print('Accuracy Score:',accuracy_score(y_test,pred))
print('Log loss : ', log_loss(y_test,pred))
print('Confusion Matrix:',confusion_matrix(y_test,pred))
print('Classification Report:','\n',classification_report(y_test,pred))
```

```
Accuracy Score: 0.9259274732620321
Log loss :  2.5583749390933366
Confusion Matrix: [[42974    30]
 [ 3516  1352]]
Classification Report:
               precision    recall  f1-score   support

           0       0.92      1.00      0.96     43004
           1       0.98      0.28      0.43      4868

    accuracy                           0.93     47872
   macro avg       0.95      0.64      0.70     47872
weighted avg       0.93      0.93      0.91     47872
```

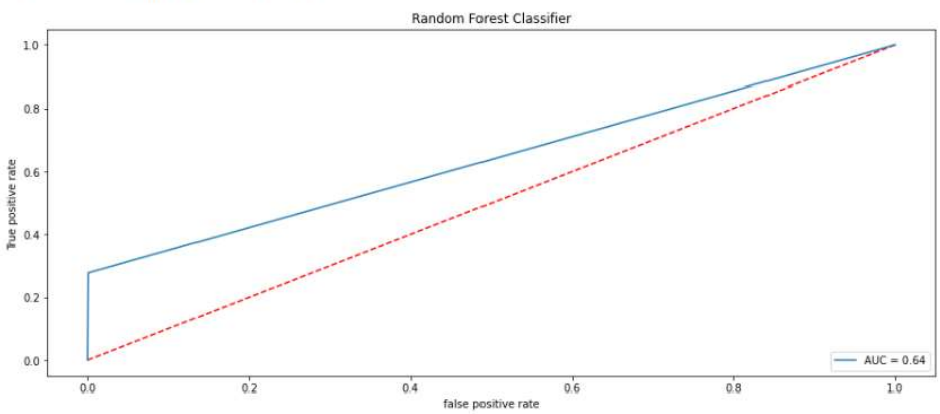
In [54]:
```
# Confusion matrix Visualization
fig, ax =plt.subplots(figsize=(5,5))
sns.heatmap(confusion_matrix(y_test, pred),annot=True,linewidths=1,center=0)
plt.xlabel("True label")
plt.ylabel("Predicted label")
bottom, top = ax.get_ylim()
ax.set_ylim(bottom + 0.5, top - 0.5)
```

Out[54]: (2.5, -0.5)



In [55]:
```
# Roc-Auc score
f,ax = plt.subplots(figsize = (15,6))
# Calculate fpr, tpr and thresholds
fpr, tpr, thresholds = roc_curve(y_test, pred)
ax.plot([0,1],[0,1],'r--')
ax.plot(fpr,tpr,label='AUC = %0.2f'% roc_auc_score(y_test, pred))
ax.legend(loc='lower right')
ax.set_xlabel('false positive rate')
ax.set_ylabel('True positive rate')
ax.set_title('Random Forest Classifier')
```

Out[55]: Text(0.5, 1.0, 'Random Forest Classifier')



In [56]:
```
def Tf_idf_test(text):
    tfid = TfidfVectorizer(max_features=43194,smooth_idf=False)
    return tfid.fit_transform(text)
```

PREDICTION

In [57]:
```
x_testing_data=Tf_idf_test(df_test['clean_comment_text'])
```

In [58]:
```
x_testing_data.shape
```

Out[58]: (153164, 43194)

In [59]:
```python
Prediction=RFC.predict(x_testing_data)
df_test['Predicted values']=Prediction
df_test
```

Out[59]:

	id	comment_text	comment_length	clean_comment_text	clean_comment_length	Predicted values
0	00001cee341fdb12	Yo bitch Ja Rule is more succesful then you'll...	367	bitch rule succesful whats hating mofuckas bit...	184	0
1	0000247867823ef7	== From RfC == \n\n The title is fine as it is...	50	title fine	10	0
2	00013b17ad220c46	" \n\n == Sources == \n\n * Zawe Ashton on Lap...	54	source zawe ashton lapland	26	0
3	00017563c3f7919a	:If you have a look back at the source, the in...	205	look source information updated correct form g...	109	0
4	00017695ad8997eb	I don't anonymously edit articles at all.	41	anonymously edit article	24	0
...	...	...	...	...	...	...
153159	fffcd0960ee309b5	. \n i totally agree, this stuff is nothing bu...	60	totally agree stuff long crap	29	0
153160	fffd7a9a6eb32c16	== Throw from out field to home plate. == \n\n...	198	throw field home plate faster throwing direct ...	85	0
153161	fffda9e8d6fafa9e	" \n\n == Okinotorishima categories == \n\n I ...	423	okinotorishima category change agree correct g...	212	0
153162	fffe8f1340a79fc2	" \n\n == ""One of the founding nations of the...	502	founding nation germany return similar israel ...	275	0
153163	ffffce3fb183ee80	" \n :::Stop already. Your bullshit is not wel...	141	stop bullshit welcome fool think kind explinat...	54	0

153164 rows × 6 columns

In [60]:
```python
df_test['Predicted values'].value_counts()
```

Out[60]:
```
0    153112
1        52
Name: Predicted values, dtype: int64
```

In [61]:
```python
df_test[df_test['Predicted values']==1].head(20)
```

Out[61]:

	id	comment_text	comment_length	clean_comment_text	clean_comment_length	Predicted values
805	0153f7856280e9ad	::::That entry made a lot of sense to me. As I...	372	entry sense replying time came desk noticed wa...	174	1
3914	06b13661ec5c3e6b	" \n\n ==Pelestinain Red Crescent Society and ...	990	pelestinain crescent society terrorism think p...	521	1
4568	07c5816cf1c0ffec	::Would you like to write up the Hegassen scro...	274	like write hegassen scroll entry publish soon ...	138	1
8358	0e02a435ccf5d6d1	== Franklin on Stalin == \n\n Possibly of inte...	382	franklin stalin possibly recently provided lin...	236	1
15183	1982942b5baedb65	'Polifacetic' isn't really an English word; th...	388	polifacetic english word entry onelook mean ve...	222	1
23370	26ffa274edf86566	==Ruud Lubbers entry== \n Hi Cary: What is hap...	485	ruud lubber entry cary happening page posted t...	219	1
25131	29e223fac14d609b	== Incorrectly titled articles by == \n\n You...	726	incorrectly titled article posted original wel...	324	1
34462	394855c528d7c0d1	== Dude == \n\n We should form a rock band. Do...	147	dude form rock band prick pissed kissed opposite	48	1
34824	39ed57532158962a	" \n\n About your Third Opinion request: The r...	385	opinion request request dispute removed declin...	239	1
36154	3c108d7fb2e8d80c	:Okay, but in 1918, the country was changed th...	278	okay 1918 country changed republic think write...	139	1
41336	44ac3a0701f504c6	" \n == Your submission at AfC Regulatory incu...	612	submission regulatory incubator accepted regul...	336	1
42507	467dbe55ed1951e8	" \n\n :::::Dude, short-term memory issues? Sc...	588	dude short term memory issue scroll page guard...	321	1
42825	47049a340480ca9b	" \n\n == Not terrible, but a bit of your own ...	1001	terrible medicine message person unknown title...	562	1
46065	4c70aff6ce8e0553	::Completely untrue. This image occurs in the...	590	completely untrue image occurs protestant chur...	415	1
47988	4fa662a56982ab54	" \n :The entries for the include ""From a mi...	274	entry include misspelling redirect misspelling...	148	1
50197	5357ea8033b3c5b3	:::You could say the same for F.B.I., but is i...	361	incorrect google show apparently feasible cons...	177	1
50338	538ca1d643b8d379	== This entry is extremely badly written! == \...	328	entry extremely badly written entry translated...	156	1
57239	5f3973189cbc083e	" \n : Good point. I've cleaned up that refere...	447	good point cleaned reference scope discussion ...	247	1
61695	66a9df620eedc33d	" \n\n ==Neutrality tag== \n This entry was ta...	674	neutrality entry tagged user march 2011 templa...	398	1
64385	6b334251852ec730	" \n *Oppose Maybe in a dedicated section titl...	166	oppose maybe dedicated section titled false fa...	99	1

In [62]:
```python
df_test.to_csv('Malignant_Predict.csv')
```

In [63]:
```python
# Pickle file.
import joblib
joblib.dump(RFC,'Malignant_Predict.pkl')
```

Out[63]: ['Malignant_Predict.pkl']

CONCLUSION

KEY FINDINGS AND CONCLUSIONS OF THE STUDY

- 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.
- From the above analysis the below mentioned results were achieved which depicts the chances and conditions of a comment being a hateful comment or a normal comment.
- With the increasing popularity of social media, more and more people consume feeds from social media and due differences they spread hate comments to instead of love and harmony. It has strong negative impacts on individual users and broader society.

LEARNING OUTCOMES OF THE STUDY IN RESPECT OF DATA SCIENCE

It is possible to classify the comments content into the required categories of Malignant and Non Malignant. However, using this kind of project an awareness can be created to know what is good and bad. It will help to stop spreading hatred among people.

LIMITATIONS OF THIS WORK AND SCOPE FOR FUTURE WORK

- Machine Learning Algorithms like Decision Tree Classifier took enormous amount of time to build the model and Ensemble techniques were taking a lot more time thus I have not included Ensemble models.
- Using Hyper-parameter tuning would have resulted in some more accuracy.
- Every effort has been put on it for perfection but nothing is perfect and this project is of no exception. There are certain areas which can be enhanced.Comment detection is an emerging research area with few public datasets. So, a lot of works need to be done on this field.