FLIP ROBO

NAME OF THE PROJECT

MALIGNANT COMMENTS CLASSIFICATION

Submitted by:

Ankita Srivastava

ACKNOWLEDGMENT

This project is based on the NLP . I research on many site but I get some useful refrences which help me to make this project.

- Wikipedia : https://en.wikipedia.org/wiki/Multi-label_classification
    - jigsaw-toxic-comment-classification-challenge
- https://www.javatpoint.com/nlp
- https://www.youtube.com/watch?v=w3coRFpyddQ&list=PLZoTAELRMXVNNrHSKv36Lr3_156yCo6Nn&ab_channel=KrishNaik
- https://stackoverflow.com/questions/35861482/nltk-lookup-error
- https://www.pythonanywhere.com/forums/topic/31724/
- https://github.com/keon/CodeGAN/issues/1

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.

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.

- ## Conceptual Background of the Domain Problem

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.

The data set contains the training set, which has approximately 1,59,000 samples and the test set which contains nearly 1,53,000 samples. All the data samples contain 8 fields which includes 'Id', 'Comments', 'Malignant', 'Highly malignant', 'Rude', 'Threat', 'Abuse' and 'Loathe'.

- ## Review of Literature

Given a number of any kind of other comments, sentences or paragraphs being used as a comment by a user, our task is to identify the comment as whether it is a malignant comment or no. After that, when we have a collection of all the malignant comments, our main task is to classify the comments into one or more categories. This problem thus comes under the category of multi-label classification problem. There is a difference between the traditional and very famous multi-class classification, and the one which we will be using, which is the multi-label classification. In a multi-class classification, each instance is classified into one of three or more classes, whereas, in a multi-label classification, multiple labels (such as ‘Comments’, ‘Malignant’, ‘Highly malignant’, ‘Rude’, ‘Threat’, ‘Abuse’ and ‘Loathe’) are to be predicted for the same instance. Multiple ways are there to approach this classification problem. It can be done using – → Multi-label methods which belong to the problem transformation category: Label Power Set (LP), Binary Relevance (BR), BR+ (BRplus), and classifier chain. 2 → Base and adapted algorithms like: (Decision Tree), Naïve Bayes, k-Nearest-Neighbor (KNN), SMO (Support Vector Machines). Further, out of the total dataset used for experimenting these algorithms, 70% was used for training and 30% was used for testing. Each testing dataset was labelled and thus for each algorithm using the predictions and labels, calculation of metric such as accuracy was done. The final results have been complied on the basis of values obtained by algorithmic models in accuracy and predicted value.

- Motivation for the Problem Undertaken

This is a huge concern as in this world, there are 7.7 billion people, and, out of these 7.7 billion, more than 3.5 billion people use some or the other form of online social media. Which means that every one-in-three people uses social media platform. This problem thus can be eliminated as it falls under the category of Natural Language Processing. In this, we try to recognize the intention of the speaker by building a model that’s capable of detecting different types of toxicity like threats, obscenity, insults, and identity-based hate. Moreover, it is crucial to handle any such kind of nuisance, to make a more user-friendly experience, only after which people can actually enjoy in participating in discussions with regard to online conversation.

Analytical Problem Framing

- Mathematical/ Analytical Modeling of the Problem

```
# 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))
```

```
[ ]  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
0.0718 ± 0.0524	fuck
0.0373 ± 0.0430	fucking
0.0295 ± 0.0298	shit
0.0201 ± 0.0172	suck
0.0194 ± 0.0211	bitch
0.0193 ± 0.0113	idiot
0.0191 ± 0.0154	stupid
0.0168 ± 0.0151	asshole
0.0116 ± 0.0121	dick
0.0114 ± 0.0106	faggot
0.0108 ± 0.0115	cunt
0.0106 ± 0.0052	gay
0.0087 ± 0.0073	hell
0.0078 ± 0.0083	ass
0.0067 ± 0.0054	bullshit
… 9985 more …	

Data Sources and their formats

Firstly I install many library because this programme need some installation like : worldcloud, nltk , xgboost, eli5.

The data set contains the training set, which has approximately 1,59,000 samples and the test set which contains nearly 1,53,000 samples. All the data samples contain 8 fields which includes ‘Id’, ‘Comments’, ‘Malignant’, ‘Highly malignant’, ‘Rude’, ‘Threat’, ‘Abuse’ and ‘Loathe’.

The label can be either 0 or 1, where 0 denotes a NO while 1 denotes a YES. There are various comments which have multiple labels. The first attribute is a unique ID associated with each comment.

The data set includes:

- **Malignant:** It is the Label column, which includes values 0 and 1, denoting if the comment is malignant or not.
- **Highly Malignant:** It denotes comments that are highly malignant and hurtful.
- **Rude:** It denotes comments that are very rude and offensive.
- **Threat:** It contains indication of the comments that are giving any threat to someone.
- **Abuse:** It is for comments that are abusive in nature.
- **Loathe:** It describes the comments which are hateful and loathing in nature.
- **ID:** It includes unique Ids associated with each comment text given.
- **Comment text:** This column contains the comments extracted from various social media platforms.

Data Preprocessing Done

Data cleaning:

I have to check what is the length of comment text..

```
[ ] 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	malignant	highly_malignant	rude	threat	abuse	loathe	length
0	0000997932d777bf	Explanation\nWhy the edits made under my usern...	0	0	0	0	0	0	264
1	000103f0d9cfb60f	D'aww! He matches this background colour I'm s...	0	0	0	0	0	0	112

```
train['clean_length'] = train.comment_text.str.len()
train.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', train.length.sum())
    print ('Clean Length', train.clean_length.sum())
```

```
Origian Length 62893130
Clean Length 43537267
```

- Data Inputs- Logic- Output Relationships

```python
# 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 parenthesis, 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))
```

- Hardware and Software Requirements and Tools Used

Hardware and Software used:--this programme used the 64-bit and 8 gb hardware.i use Google colab for perform this project because it need very fast response.it take too much time for run.

```
!pip install wordcloud
!pip3 install xgboost

!pip install nltk

!pip install eli5
```

These are software which I use for run this..and many other software I use for run this programme like:

```
[ ] import nltk
    nltk.download('stopwords')

    [nltk_data] Downloading package stopwords to /root/nltk_data...
    [nltk_data]   Unzipping corpora/stopwords.zip.
    True
```

```
[ ] import nltk
    nltk.download('wordnet')

    [nltk_data] Downloading package wordnet to /root/nltk_data...
    True
```

```
[ ] import nltk
    nltk.download('omw-1.4')

    [nltk_data] Downloading package omw-1.4 to /root/nltk_data...
    True
```

Model/s Development and Evaluation

- Identification of possible problem-solving approaches (methods)

```
[ ] #  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
```

.

- Testing of Identified Approaches (Algorithms)

```
[ ] 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
```

- Run and Evaluate selected models

```
# 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 0.9552139037433155
[[42729   221]
 [ 1923  2999]]
              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 0.9988092999937331
Test accuracy is 0.9401111296791443
[[41621  1329]
 [ 1538  3384]]
              precision    recall  f1-score   support

           0       0.96      0.97      0.97     42950
           1       0.72      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 0.9988092999937331
Test accuracy is 0.9552347927807486
[[42421   529]
 [ 1614  3308]]
              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
```

```python
# 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 0.9396234523138076
Test accuracy is 0.9366226604278075
[[42846   104]
 [ 2930  1992]]
              precision    recall  f1-score   support

           0       0.94      1.00      0.97     42950
           1       0.95      0.40      0.57      4922

    accuracy                           0.94     47872
   macro avg       0.94      0.70      0.77     47872
weighted avg       0.94      0.94      0.92     47872
```

```python
#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 0.9510738681635467
Test accuracy is 0.9489263034759359
[[42553   397]
 [ 2048  2874]]
              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))
```

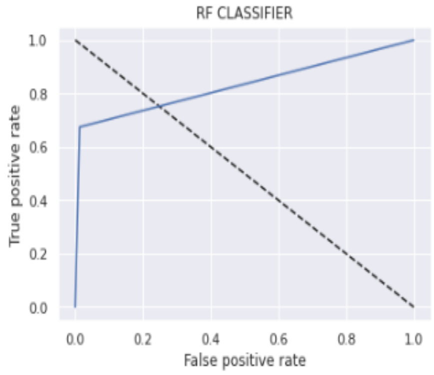
```
Training accuracy is 0.9223627785387515
Test accuracy is 0.917697192513369
[[42812   138]
 [ 3802  1120]]
              precision    recall  f1-score   support

           0       0.92      1.00      0.96     42950
           1       0.89      0.23      0.36      4922

    accuracy                           0.92     47872
   macro avg       0.90      0.61      0.66     47872
weighted avg       0.92      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 0.9987913947304811
Test accuracy is 0.9549841243315508
cross validation score : 95.65835811736271
[[42398   552]
 [ 1603  3319]]
              precision    recall  f1-score   support

           0       0.96      0.99      0.98     42950
           1       0.86      0.67      0.75      4922

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

In all the metrics only Random Forest Classifier give highest accuracy.So I plot the AUC curve for this

```python
#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()
```

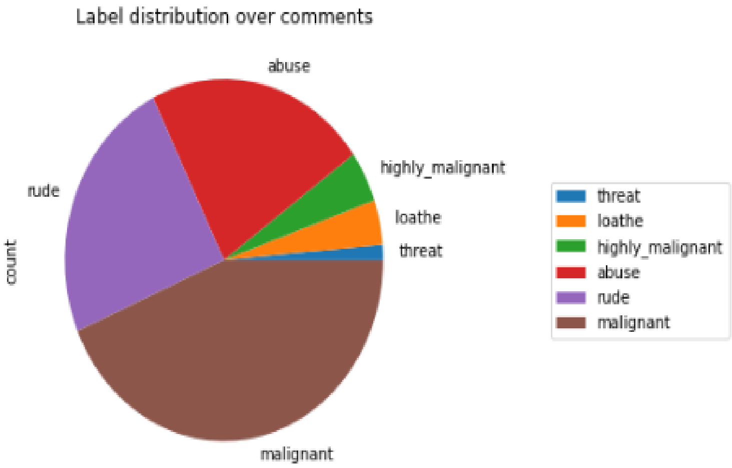


Here we can see that More area are under the Curve.

Visualizations

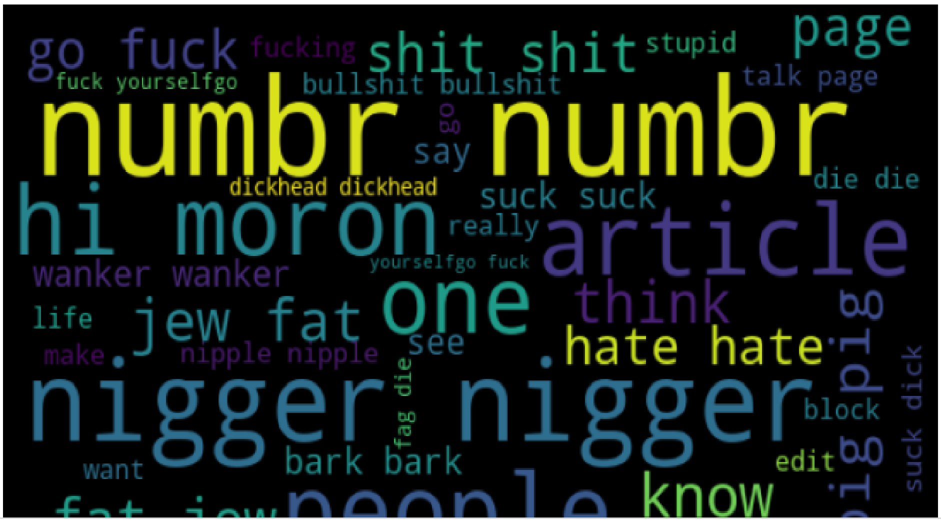
```python
cols_target = ['malignant','highly_malignant','rude','threat','abuse','loathe']
df_distribution = train[cols_target].sum()\
                            .to_frame()\
                            .rename(columns={0: 'count'})\
                            .sort_values('count')

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

```
<matplotlib.legend.Legend at 0x7fa416ec6190>
```



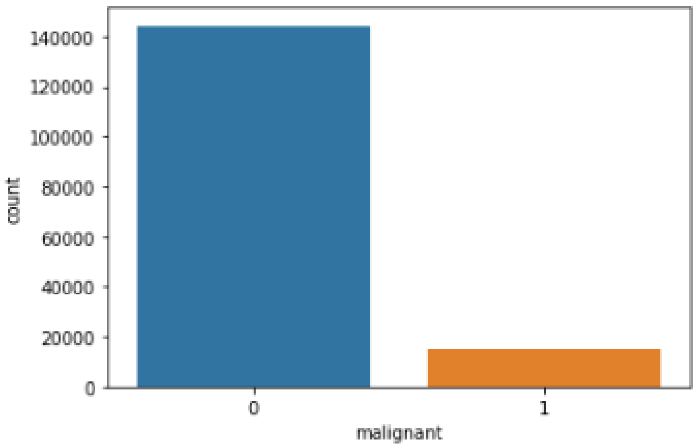
Here we can see that Malignant comment ratio is high, so we should

```python
#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()
```




```python
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
```



- Interpretation of the Results

The following steps were taken to process the data: →

- A string without all punctuations to be prepared
- Stop words are those words that are frequently used in both written and verbal communication and thereby do not have either a positive or negative impact on our statement like "is, this, us, etc."
- Lemmatizing is the process of grouping together the inflected forms of a word so they can be analyzed as a single item.

CONCLUSION

- Key Findings and Conclusions of the Study

```
test_data =tf_vec.fit_transform(test['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
```

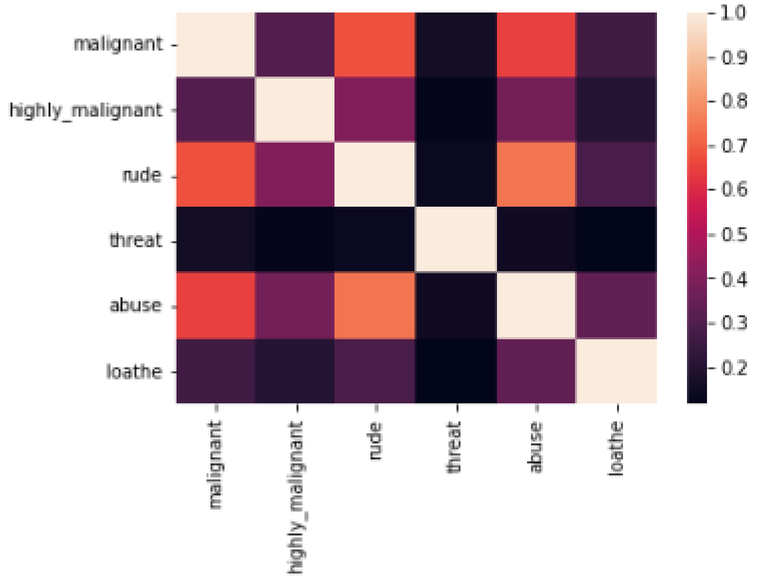
```
array([0, 0, 0, ..., 1, 0, 0])
```

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

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

```
[ ]  print(sns.heatmap(train.corr()))
```

```
AxesSubplot(0.125,0.125;0.62x0.755)
```



- ## Learning Outcomes of the Study in respect of Data Science

This is the summary of the things we followed in this project:

1..The first step involved collecting data and deciding what part of it is suitable for training : This step was extremely crucial since including only very small length comments would give poor results.

2..The second major step was performing cleaning of data including punctuation removal, stop word removal, stemming and lemmatizing.

3..The third step was choosing models to train on: Since I had a wide variety of models, selecting which all models to train and test took lots of efforts.

4.. Finally comparing on the basis of different evaluation metrics.

- ## Limitations of this work and Scope for Future Work

The current project predicts the type or offensive word in the comment. We are planning to add the following features in the future:

→ Analyse which age group is being toxic towards a particular group or brand.

→ Add feature to automatically sensitize words which are classified as toxic.

→ Automatically send alerts to the concerned authority if threats are classified as severe.

→ Build a feedback loop to further increase the efficiency of the model.

→ Handle mistakes and short forms of words to get better accuracy of the result.