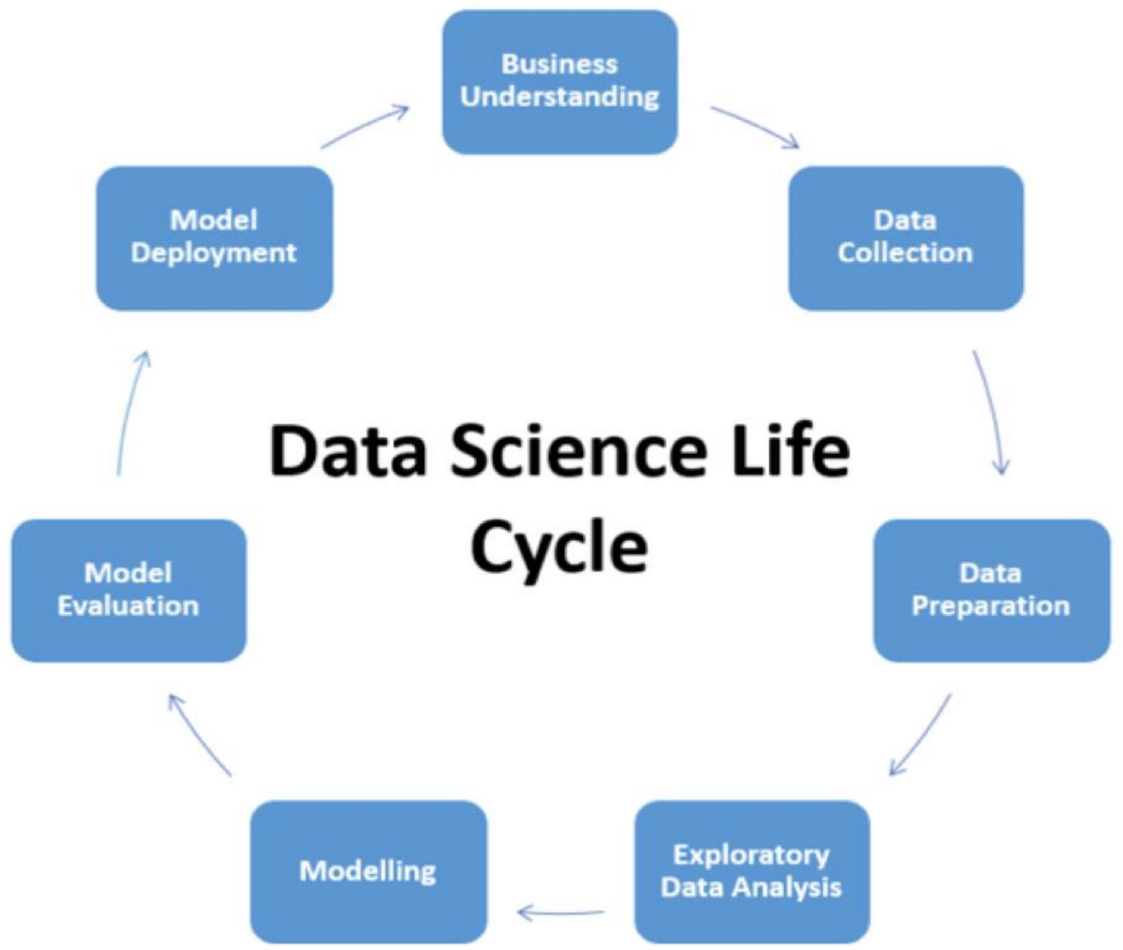
MALIGNANT COMMENTS CLASSIFICATION

Harsh Raj Gupta

USE CASE:

- 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 behaviour.
- 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.

APPROACH AND LIFE CYCLE :



DATA DESCRIPTION :

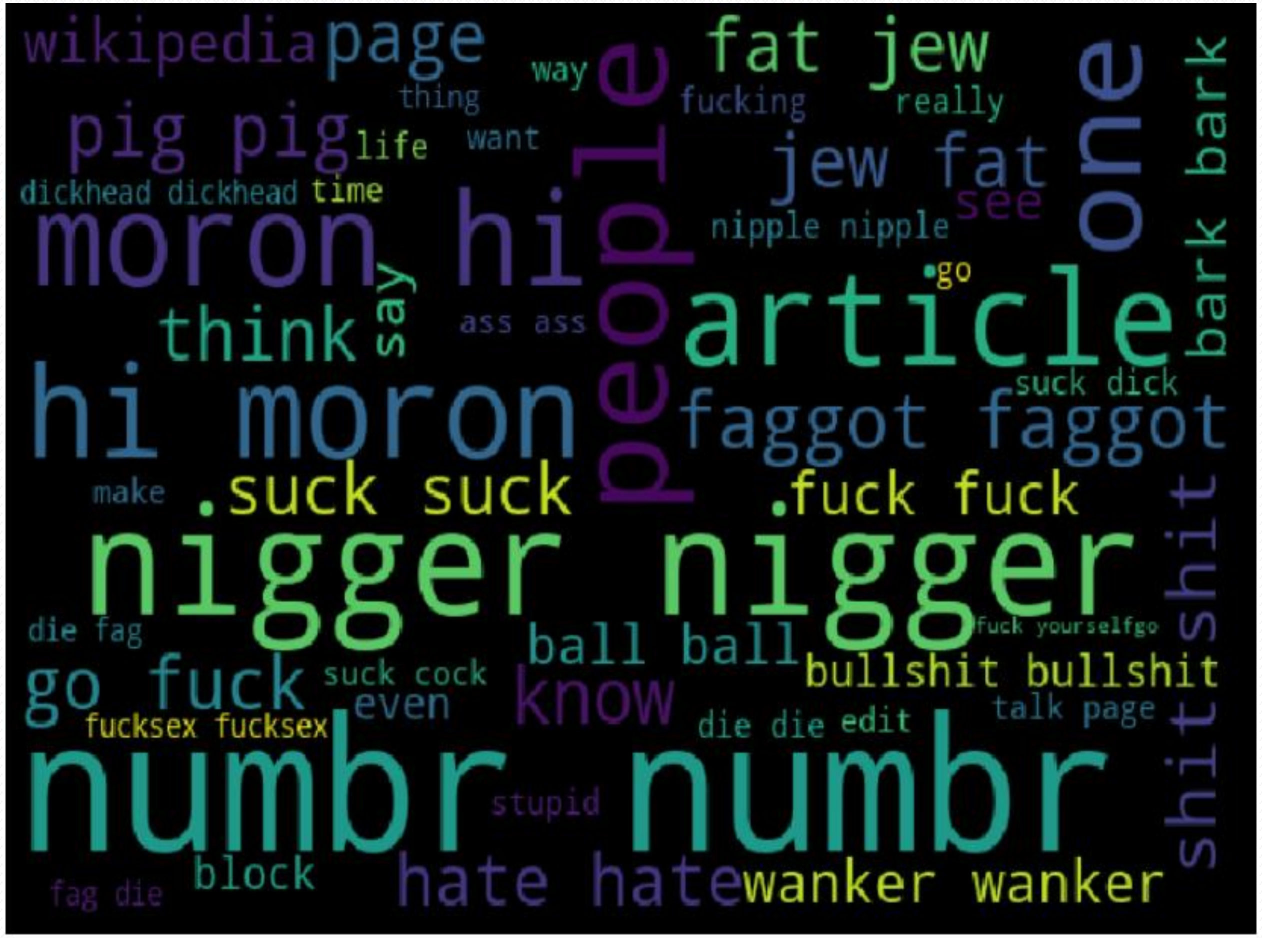
- Project contain train and test dataset.
- In train data set there are 159,571 rows and 8 columns.
- In test data set it is like 153,164 rows and 2 columns.
- There are no null values in the dataset
- Most of the data are numeric in nature which are binary.
- Comments is object in nature and consist of text.
- Overall memory usage for train and test is around 15MB.

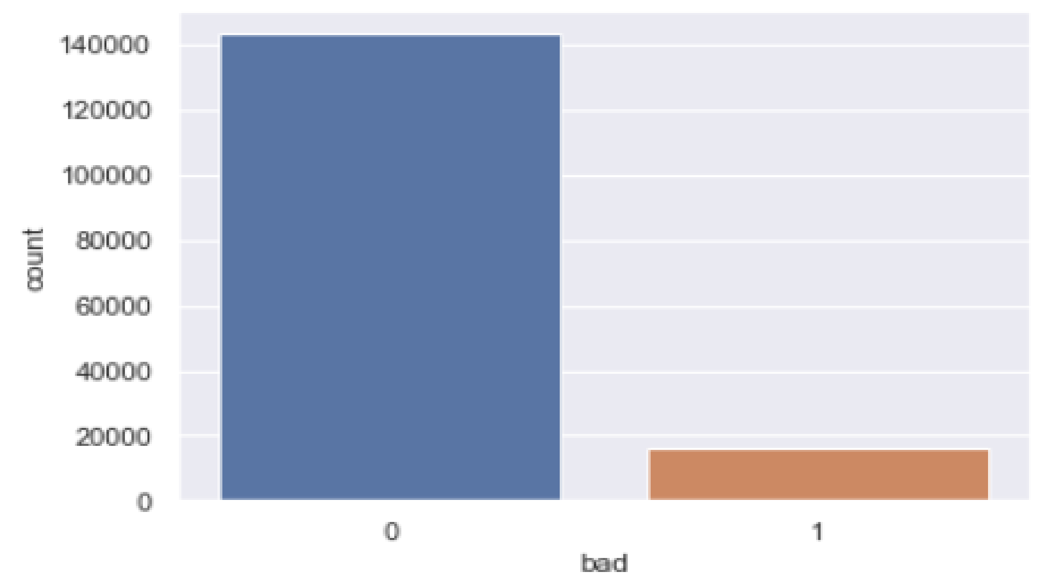
DATA PRE-PROCESSING :

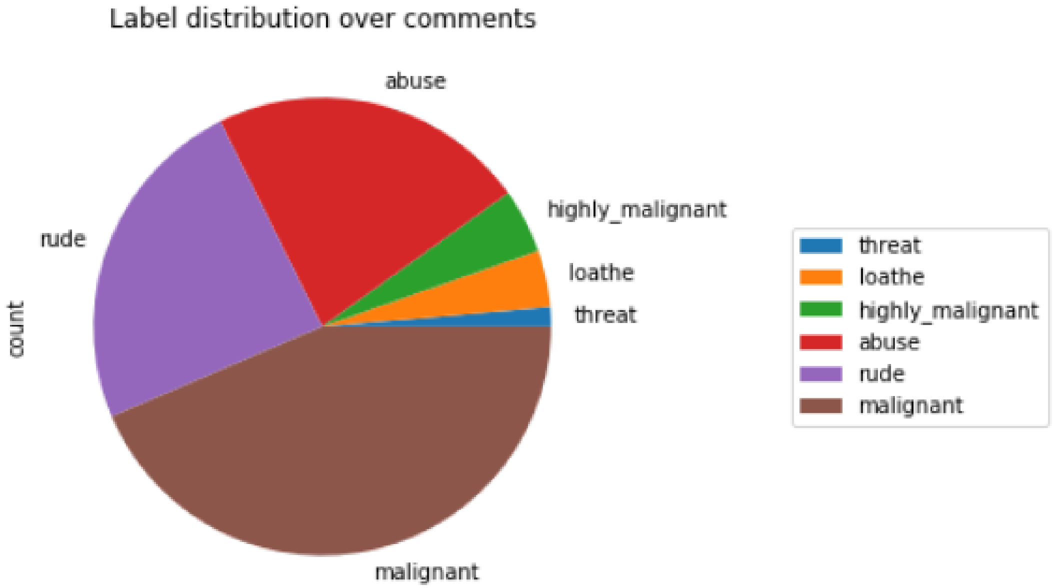
AS THERE ARE NO NULL VALUES IN THE DATASET, BUT AS THE COMMENT COLUMN IN TEXT FORMAT SO THERE REQUIRE LOT OF TEXT PRE-PROCESSING.

```
# Convert all messages to lower case
train['comment_text'] = train['comment_text'].str.lower()

# Replace email addresses with 'email'
train['comment_text'] = train['comment_text'].str.replace(r'^.+@[^\.].*\.[a-z]{2,}$',
                                 'emailaddress')

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

# Replace money symbols with 'moneysymb' (£ can by typed with ALT key + 156)
train['comment_text'] = train['comment_text'].str.replace(r'£|\$', 'dollers')

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


# Replace numbers with 'numbr'
train['comment_text'] = train['comment_text'].str.replace(r'\d+(\.\d+)?', 'numbr')


train['comment_text'] = train['comment_text'].apply(lambda x: ' '.join(
    term for term in x.split() if term not in string.punctuation))

stop_words = set(stopwords.words('english') + ['u', 'ü', 'ur', '4', '2', 'im', 'dont', 'doin', 'ure'])
train['comment_text'] = train['comment_text'].apply(lambda x: ' '.join(
    term for term in x.split() if term not in stop_words))

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

EDA :


wikipedia page
way
fat jew
people
one
bark bark
thing
fucking
really
pig pig
life
want
jew fat
dickhead dickhead
time
see
nipple nipple
moron hi
go
say
think
ass ass
article
bark bark
suck dick
hi moron
faggot faggot
make
.suck suck
.fuck fuck
shitshit
nigger nigger
die fag
fuck yourselfgo
ball ball
go fuck
suck cock
bullshit bullshit
know
even
talk page
fucksex fucksex
die die edit
numbr numbr
stupid
block
fag die
hate hate
wanker wanker


140000
120000
100000
80000
60000
40000
20000
0
count
0
1
bad


Label distribution over comments
abuse
highly_malignant
rude
loathe
threat
malignant
count
threat
loathe
highly_malignant
abuse
rude
malignant

MODEL BUILDING :

- The **model building** process involves setting up ways of collecting **data**, understanding and paying attention to what is important in the **data** to answer the questions you are asking, finding a statistical, mathematical or a simulation **model** to gain understanding and make predictions.

Evaluation Matrices:

- **Accuracy** - it determines how often a model predicts default and non default correctly.
- **Precision**-it calculates whenever our models predicts it is default how often it is correct.
- **Recall**- Recall regulate the actual default that the model is actually predict.
- **Precision Recall Curve** - PRC will display the tradeoff between Precision and Recall threshold.
- **F1 score** - the F1-score, is a measure of a model's accuracy on a dataset. It is used to evaluate binary classification systems, which classify examples into 'positive' or 'negative'.

Cross Validations:

RANDOM FOREST CLASSIFIER GIVING BEST RESULTS AMONGST ALL ALGORITHMS :

```
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 0.9988719684151156
Test accuracy is 0.9551512366310161
cross validation score : 95.6708914874724
[[42416   534]
 [ 1613  3309]]
              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
```

Random forest, like its name implies, consists of a large number of individual decision trees that operate as an ensemble. Each individual tree in the random forest spits out a class prediction and the class with the most votes becomes our model's prediction

Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean/average prediction (regression) of the

ROC-CURVE :

```
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()
```

- An **ROC curve** (**receiver operating characteristic curve**) is a **graph** showing the performance of a classification model at all classification thresholds. This **curve** plots two parameters: True Positive Rate. False Positive Rate

  - Most of area is lying under the curve and providing the high true positive rate approx. 85% which is good sign of better prediction

CONCLUSIONS :

- In this project there are some variables like malignant and rude which are highly correlated it is possible because one comment text may have combination of multiple features.
- There were no null values in the data set only the pre processing is required.
- Removing the column id does not impact the model training.
- Using decision tree, model can reduce the false negative values
- It has future scope in various use cases likewise in election, social media etc, where every day there are multi offensive comments spread.
- Random forest is well suitable for this project as it used tree internally and it used multiple weak learner and generate the strong model and generate low bias and low variance model.