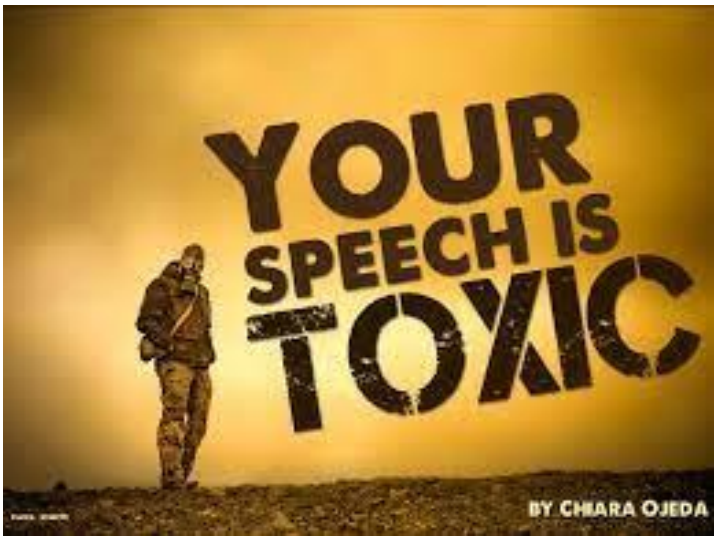
FLIP ROBO

Malignant Comments Classifier Project



Submitted by:

Subhashchandra Pandey

ACKNOWLEDGMENT

I would like to express my sincere gratitude to my SME Mr. Shwetank Mishra from Flip Robo Technologies for letting me work on this project and providing me with all necessary information and dataset for the project. I would also like to thank my mentor of Data Trained academy for providing me sufficient knowledge so that I can complete the project.

I have utilized a few external resources that helped me to complete the project. I ensured that I learn from the samples and modify things according to my project requirement. All the external resources that were used in creating this project are listed below:

- https://www.google.com/
- https://www.youtube.com/
- https://scikit-learn.org/stable/user_guide.html
- https://github.com/
- https://www.analyticsvidhya.com/

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 behaviour.

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

Online platforms and social media become the place where people share the thoughts freely without any partiality and overcoming all

the race people share their thoughts and ideas among the crowd. Social media is a computer-based technology that facilitates the sharing of ideas, thoughts, and information through the building of virtual networks and communities. By design, social media is Internet-based and gives users quick electronic communication of content. Content includes personal information, documents, videos, and photos. Users engage with social media via a computer, tablet, or smartphone via web-based software or applications.
While social media is ubiquitous in America and Europe, Asian countries like India lead the list of social media usage. More than 3.8 billion people use social media.

In this huge online platform or an online community there are some people or some motivated mob wilfully bully others to make them not to share their thought in rightful way. They bully others in a foul language which among the civilized society is seen as ignominy. And when innocent individuals are being bullied by these mob these individuals are going silent without speaking anything. So, ideally the motive of this disgraceful mob is achieved.

To solve this problem, we are now building a model that identifies all the foul language and foul words, using which the online platforms like social media principally stops these mob using the foul language in an online community or even block them or block them from using this foul language.

- Review of Literature

The purpose of the literature review is to:
1. Identify the foul words or foul statements that are being used.
2. Stop the people from using these foul languages in online public forum.
To solve this problem, we are now building a model using our machine language technique that identifies all the foul language and foul words, using which the online platform like social media principally stops these mob using the foul language in an online

community or even block them or block them from using this foul language.
I have used 8 different Classification algorithms and shortlisted the best on basis of the metrics of performance and I have chosen one algorithm and build a model in that algorithm.
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.
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.

- Motivation for the Problem Undertaken

The main objective of this study is to investigate which method from a chosen set of machine learning techniques performs the best. So far, we have a range of publicly available models served through the Perspective API, including toxicity/malignant comments. But the current models still make errors, and they don't allow users to select which type of toxicity they are interested in finding.
The project which is given to us as a part of the internship programme which gives an insight to identify major factors that lead to cyberbullying and online abusive comments. 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 was to classify the news to bring awareness and reduce unwanted chaos and make a good model which will help us to know such kind of miscreants. 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.

Analytical Problem Framing

- Mathematical/ Analytical Modelling of the Problem

We are provided with two different datasets. One for training and another one to test the efficiency of the model created using the training dataset. The training data provided here has both dependent and independent variables. As it is a multiclass problem it has 6 independent/target variables. Here the target variables named “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.
Clearly it is a binary classification problem as the target columns giving binary outputs and all independent variables has text so it is a supervised machine learning problem where we can use the techniques of NLP and classification-based algorithms of Machine Learning. Here we will use NLP techniques like word tokenization, lemmatization, stemming and tfidf vectorizer then those processed data will be used to create best model using various classification based supervised ML algorithms like Logistic Regression, Multinomial NB, LGBM Classifier, XGB Classifier, Gradient Boosting Classifier, LinearSVC, Decision Tree Classifier and Adaboost Classifier.

- Data Sources and their formats

Data set provided by Flip Robo was in the format of CSV (Comma Separated Values). The data set contains the training set, which has approximately 159571 samples and the test set which contains nearly 153164 samples. All the data samples contain 8 fields which includes ‘Id’, ‘Comments’, ‘Malignant’, ‘Highly malignant’, ‘Rude’, ‘Threat’, ‘Abuse’ and ‘Loathe’. In the dataset all the columns are of object data type. The attribution information is as follows:

Variables	Definition
id	It includes unique Ids associated with each comment text given
comment_text	The comments extracted from various social media platforms
malignant	It denotes the comments are 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

- Data Preprocessing Done

Data pre-processing is the process of converting raw data into a well-readable format to be used by Machine Learning model. Data pre-processing is an integral step in Machine Learning as the quality of data and the useful information that can be derived from it directly affects the ability of our model to learn; therefore, it is extremely important that we pre-process our data before feeding it into our model. I have used following pre-processing steps:

➢ Importing necessary libraries and loading dataset as a data frame.

➢ Checked some statistical information like shape, number of unique values present, info, null values, value counts, duplicated values etc.

➢ Checked for null values and did not find any null values. And removed Id.

➢ Done feature engineering and created new columns viz label: which contain both good and bad comments which is the sum of all the labels,

comment_length: which contains the length of comment text.

➢ Visualized each feature using seaborn and matplotlib libraries by plotting categorical plots like pie plot, count plot, distribution plot and wordcloud for each label.

➢ Done text pre-processing techniques like Removing Punctuations and other special characters, Splitting the comments into individual words, Removing Stop Words, Stemming and Lemmatization. Then created new column as clean_length after cleaning the data. All these steps were done on both train and test datasets. Checked correlation using heatmap.

➢ After getting a cleaned data used TF-IDF vectorizer. It'll help to transform the text data to feature vector which can be used as input in our modelling. It is a common algorithm to transform text into numbers. It measures the originality of a word by comparing the frequency of appearance of a word in a document with the number of documents the words appear in. Mathematically,

TF-IDF = TF(t*d)*IDF(t,d)

➢ Balanced the data using Randomoversampler method.

- Data Inputs- Logic- Output Relationships

The train dataset consists of multilabel and features. The features are independent, and label is dependent as the values of our independent variables changes as our label varies.

• I checked the distribution of skewness using dist plots and used count plots to check the counts available in each column as a part of univariate analysis.

• Got to know sense of loud words in every label using word cloud which gives the words frequented in the labels.

• I have checked the correlation between the label and features using heat map.

- Hardware and Software Requirements and Tools Used

Hardware Used:

1. Processor — core i5 and above
2. RAM — 8 GB or above
3. SSD — 250GB or above

Software Used:

i. Programming language: Python
ii. Distribution: Anaconda Navigator
iii. Browser based language shell: Jupyter Notebook
Libraries/Packages Used:
Pandas, NumPy, matplotlib, seaborn, scikit-learn and pandas_profiling

Model/s Development and Evaluation

- Identification of possible problem-solving approaches (methods)

In this project there were 6 features which defines the type of comment like malignant, hate, abuse, threat, loathe but we created another feature named as “label” which is combined of all the above features and contains the labelled data into the format of 0 and 1 where 0 represents “NO” and 1 represents “Yes”. In this NLP based project we need to predict the multiple labels which are binary. I have converted text into feature vectors using TF-IDF vectorizer and separated our feature and labels. Also, before building the model, I made sure that the input data is cleaned and scaled before it was fed into the machine learning models.

- Testing of Identified Approaches (Algorithms)

Since the target variable is categorical in nature, from this I can conclude that it is a classification type problem hence I have used following classification algorithms. After the pre-processing and data cleaning I left with 10 columns including targets. The algorithms used on training the data are as follows:

1. Logistic Regression
2. MultinomialNB
3. LightGBM Classifier
4. LinearSVC
5. Gradient Boosting Classifier
6. Decision Tree Classifier
7. Extreme Gradient Boosting Classifier (XGB)
8. AdaBoost Classifier

- Run and Evaluate selected models

I have used 8 classification algorithms after choosing random state as 42. First, I have created 8 different classification algorithms and are appended in the variable models. Then, ran a for loop which contained the accuracy of the models along with different evaluation metrics.

```
# Creating instances for different Classifiers

LR = LogisticRegression()
MNB = MultinomialNB()
lgbm = LGBMClassifier()
GB = GradientBoostingClassifier()
SVC = LinearSVC()
DTC = DecisionTreeClassifier()
ABC = AdaBoostClassifier()
xgb = XGBClassifier(verbosity=0)

# 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(('LGBMClassifier',lgbm))
models.append(('GradientBoostingClassifier',GB))
models.append(('LinearSVC',SVC))
models.append(('DecisionTreeClassifier',DTC))
models.append(('AdaBoostClassifier',ABC))
models.append(('XGBClassifier',xgb))
```

```
# Creating empty lists
Model=[]
Score=[]
Acc_score=[]
cvs=[]
rocscore=[]
lg_loss=[]
Hamming_loss=[]

for name,model in models:
    print("**********",name,"**********")
    print("\n")
    Model.append(name)
    model.fit(train_x,train_y)
    print(model)
    y_pred=model.predict(x_test)
# Accuracy Score
    acc_score=accuracy_score(y_test,y_pred)
    print('Accuracy_Score: ',acc_score)
    Acc_score.append(acc_score*100)
# Model Score
    score=model.score(train_x,train_y)
    print('Learning Score : ',score)
    Score.append(score*100)
# Cross Validation Score
    cv=cross_val_score(model,X,y,cv=5,scoring='accuracy').mean()
    print('Cross Validation Score: ',cv)
    cvs.append(cv*100)
```

```
# Auc Roc Score
    roc_auc= roc_auc_score(y_test,y_pred)
    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)
# Hamming Loss
    ham_loss = hamming_loss(y_test,y_pred)
    print("Hamming loss: ", ham_loss)
    Hamming_loss.append(ham_loss)
    print('\n')
# Confusion Matrix
    print('Confusion matrix: \n')
    cm=confusion_matrix(y_test,y_pred)
    print(cm)
    print("\n")
# Classification Report
    print('Classification Report:\n ')
    print(classification_report(y_test,y_pred))
    print("*******************************************************************************")
    print('\n\n')
```

```
********** LogisticRegression **********


LogisticRegression()
Accuracy_Score:  0.9457093917112299
Learning Score :  0.9514231369377784
Cross Validation Score:  0.9559694387031117
roc_auc_score:  0.8959121400260334
Log loss :  1.8751610446728375
Hamming loss:  0.054290608288770054


Confusion matrix:

[[41216  1788]
 [  811  4057]]


Classification Report:

              precision    recall  f1-score   support

           0       0.98      0.96      0.97     43004
           1       0.69      0.83      0.76      4868

    accuracy                           0.95     47872
   macro avg       0.84      0.90      0.86     47872
weighted avg       0.95      0.95      0.95     47872

************************************************************
********** MultinomialNB **********


MultinomialNB()
Accuracy_Score:  0.9107202540106952
Learning Score :  0.9147085957698835
Cross Validation Score:  0.9463499000343936
roc_auc_score:  0.885090259704874
Log loss :  3.0836726119548987
Hamming loss:  0.08927974598930481


Confusion matrix:

[[39446  3558]
 [  716  4152]]


Classification Report:

              precision    recall  f1-score   support

           0       0.98      0.92      0.95     43004
           1       0.54      0.85      0.66      4868

    accuracy                           0.91     47872
   macro avg       0.76      0.89      0.80     47872
weighted avg       0.94      0.91      0.92     47872

************************************************************
```

********** LGBMClassifier **********

LGBMClassifier()
Accuracy_Score: 0.9477147393048129
Learning Score : 0.9059043952664609
Cross Validation Score: 0.95548689342935
roc_auc_score: 0.8672435985966641
Log loss : 1.8058917272508708
Hamming loss: 0.052285260695187165

Confusion matrix:

[[41639 1365]
 [1138 3730]]

Classification Report:

	precision	recall	f1-score	support
0	0.97	0.97	0.97	43004
1	0.73	0.77	0.75	4868
accuracy			0.95	47872
macro avg	0.85	0.87	0.86	47872
weighted avg	0.95	0.95	0.95	47872

****
********** GradientBoostingClassifier **********

GradientBoostingClassifier()
Accuracy_Score: 0.9440382687165776
Learning Score : 0.8277599972664836
Cross Validation Score: 0.9402209667134661
roc_auc_score: 0.7923294689599875
Log loss : 1.9328621002603092
Hamming loss: 0.05596173128342246

Confusion matrix:

[[42263   741]
 [1938 2930]]

Classification Report:

	precision	recall	f1-score	support
0	0.96	0.98	0.97	43004
1	0.80	0.60	0.69	4868
accuracy			0.94	47872
macro avg	0.88	0.79	0.83	47872
weighted avg	0.94	0.94	0.94	47872

****

```
********** LinearSVC **********


LinearSVC()
Accuracy_Score:  0.9397560160427807
Learning Score :  0.9713037733914965
Cross Validation Score:  0.9592407120377594
roc_auc_score:  0.884491951103209
Log loss :  2.0807866294331543
Hamming loss:  0.06024398395721925


Confusion matrix:

[[41020  1984]
 [  900  3968]]


Classification Report:

              precision    recall  f1-score   support

           0       0.98      0.95      0.97     43004
           1       0.67      0.82      0.73      4868

    accuracy                           0.94     47872
   macro avg       0.82      0.88      0.85     47872
weighted avg       0.95      0.94      0.94     47872

************************************************************

 ********** DecisionTreeClassifier **********


 DecisionTreeClassifier()
 Accuracy_Score:  0.9288101604278075
 Learning Score :  0.9981947402590007
 Cross Validation Score:  0.9408852466783897
 roc_auc_score:  0.8346788095689939
 Log loss :  2.458843823893802
 Hamming loss:  0.0711898395721925


 Confusion matrix:

 [[40976  2028]
  [ 1380  3488]]


 Classification Report:

               precision    recall  f1-score   support

           0       0.97      0.95      0.96     43004
           1       0.63      0.72      0.67      4868

    accuracy                           0.93     47872
   macro avg       0.80      0.83      0.82     47872
weighted avg       0.93      0.93      0.93     47872

 ************************************************************************
```

```
********** AdaBoostClassifier **********


AdaBoostClassifier()
Accuracy_Score:  0.9267003676470589
Learning Score :  0.8445939019806604
Cross Validation Score:  0.9457733566448472
roc_auc_score:  0.8172003280809254
Log loss :  2.531712182167665
Hamming loss:  0.07329963235294118


Confusion matrix:

[[41054  1950]
 [ 1559  3309]]


Classification Report:

              precision    recall  f1-score   support

           0       0.96      0.95      0.96     43004
           1       0.63      0.68      0.65      4868

    accuracy                           0.93     47872
   macro avg       0.80      0.82      0.81     47872
weighted avg       0.93      0.93      0.93     47872

************************************************************
 ********** XGBClassifier **********


 XGBClassifier(base_score=0.5, booster='gbtree', callbacks=None,
               colsample_bylevel=1, colsample_bynode=1, colsample_bytree=1,
               early_stopping_rounds=None, enable_categorical=False,
               eval_metric=None, gamma=0, gpu_id=-1, grow_policy='depthwise',
               importance_type=None, interaction_constraints='',
               learning_rate=0.300000012, max_bin=256, max_cat_to_onehot=4,
               max_delta_step=0, max_depth=6, max_leaves=0, min_child_weight=1,
               missing=nan, monotone_constraints='()', n_estimators=100,
               n_jobs=0, num_parallel_tree=1, predictor='auto', random_state=0,
               reg_alpha=0, reg_lambda=1, ...)
 Accuracy_Score:  0.9493440842245989
 Learning Score :  0.9115536623423957
 Cross Validation Score:  0.9536005912254529
 roc_auc_score:  0.8550342854732054
 Log loss :  1.7496124393700108
 Hamming loss:  0.05065591577540107


 Confusion matrix:

 [[41861  1143]
  [ 1282  3586]]


 Classification Report:

               precision    recall  f1-score   support

           0       0.97      0.97      0.97     43004
           1       0.76      0.74      0.75      4868

    accuracy                           0.95     47872
   macro avg       0.86      0.86      0.86     47872
weighted avg       0.95      0.95      0.95     47872

 ************************************************************************
```

Model Selection:

```
# Displaying Scores and metrics:
Results=pd.DataFrame({'Model': Model,'Learning Score': Score,'Accuracy Score': Acc_score,'Cross Validation Score':cvs,
                      'Auc_Roc_Score':rocscore,'Log_Loss':lg_loss,'Hamming_loss':Hamming_loss})
Results
```

	Model	Learning Score	Accuracy Score	Cross Validation Score	Auc_Roc_Score	Log_Loss	Hamming_loss
0	LogisticRegression	95.142314	94.570939	95.596944	89.591214	1.875161	0.054291
1	MultinomialNB	91.470860	91.072025	94.634990	88.509026	3.083673	0.089280
2	LGBMClassifier	90.590440	94.771474	95.548689	86.724360	1.805892	0.052285
3	GradientBoostingClassifier	82.776000	94.403827	94.022097	79.232947	1.932862	0.055962
4	LinearSVC	97.130377	93.975602	95.924071	88.449195	2.080787	0.060244
5	DecisionTreeClassifier	99.819474	92.881016	94.088525	83.467881	2.458844	0.071190
6	AdaBoostClassifier	84.459390	92.670037	94.577336	81.720033	2.531712	0.073300
7	XGBClassifier	91.155366	94.934408	95.360059	85.503429	1.749612	0.050656

After creating and training different classification algorithms, we can see that the difference between accuracy and cross validation score is less for "Extreme Gradient Boosting Classifier (XGBClassifier)" and “Gradient Boosting Classifier”. But, “XGBClassifier” giving less loss values, high auc roc score and accuracy score compared to Gradient Boosting Classifier. On this basis I can conclude that "XGBClassifier" as the best fitting model. Now, we will try Hyperparameter Tuning to find out the best parameters and using them to improve the scores and metrics values.

Hyper Parameter Tuning:

```
# Let's Use the GridSearchCV to find the best paarameters in XGBClassifier

# Extreme XGBClassifier
parameters = {'n_estimators':[100,1000],
              'booster':['gbtree'],
              'max_depth':[2,6],
              'eta':[0,0.2,0.3],
              'colsample_bytree':[1,0.8]}

# Running GridSearchCV for the model Bagging Regressor.
GCV=GridSearchCV(XGBClassifier(),parameters,cv=5,scoring='accuracy')
```

```
# Training the best model
GCV.fit(train_x,train_y)
```

```
GridSearchCV(cv=5,
             estimator=XGBClassifier(base_score=None, booster=None,
                                     colsample_bylevel=None,
                                     colsample_bynode=None,
                                     colsample_bytree=None, gamma=None,
                                     gpu_id=None, importance_type='gain',
                                     interaction_constraints=None,
                                     learning_rate=None, max_delta_step=None,
                                     max_depth=None, min_child_weight=None,
                                     missing=nan, monotone_constraints=None,
                                     n_estimators=100, n_jobs=None,
                                     num_parallel_tree=None, random_state=None,
                                     reg_alpha=None, reg_lambda=None,
                                     scale_pos_weight=None, subsample=None,
                                     tree_method=None, validate_parameters=None,
                                     verbosity=None),
             param_grid={'booster': ['gbtree'], 'colsample_bytree': [1, 0.8],
                         'eta': [0, 0.2, 0.3], 'max_depth': [2, 6],
                         'n_estimators': [100, 1000]},
             scoring='accuracy')
```

```
#Getting best parameters
GCV.best_params_
```

```
{'booster': 'gbtree',
 'colsample_bytree': 1,
 'eta': 0.3,
 'max_depth': 6,
 'n_estimators': 1000}
```

I have used 5 XGBClassifier parameters to be saved under the variable "parameters" that will be used in GridSearchCV for finding the best output. Assigned a variable to the GridSearchCV function after entering all the necessary inputs. And we used our training data set to make the GridSearchCV aware of all the hyper parameters that needs to be applied on our best model.

Creating Final Model:

```
# Creating final model
comment_model = XGBClassifier(n_estimators=1000, max_depth=6, eta=0.3, colsample_bytree=1, booster='gbtree')
comment_model.fit(train_x, train_y)
pred = comment_model.predict(x_test)
acc_score = accuracy_score(y_test,pred)
print("Accuracy score:", acc_score*100)
roc_auc = roc_auc_score(y_test,y_pred)
print('roc_auc_score: ',roc_auc*100)
print('Log loss : ', log_loss(y_test,pred))
print("Hamming loss: ", hamming_loss(y_test,pred))
print("\n")
print('Confusion Matrix: \n',confusion_matrix(y_test,pred))
print('\n')
print('Classification Report:','\n',classification_report(y_test,pred))
```

```
Accuracy score: 95.48587901069519
roc_auc_score:  85.50342854732054
Log loss :  1.5591399264928922
Hamming loss:  0.04514120989304813


Confusion Matrix:
 [[41940  1064]
 [ 1097  3771]]


Classification Report:
              precision    recall  f1-score   support

           0       0.97      0.98      0.97     43004
           1       0.78      0.77      0.78      4868

    accuracy                           0.95     47872
   macro avg       0.88      0.87      0.88     47872
weighted avg       0.95      0.95      0.95     47872
```

I have successfully incorporated the hyper parameter tuning using best parameters of XGBClassifier and the accuracy of the model has been increased after hyperparameter tuning and received the accuracy score as 95.48% which is very good.

Saving the final model and predicting the results

```
# Saving the model using .pkl
import joblib
joblib.dump(comment_model,"Malignant_Comments_Classification.pkl")
```

```
['Malignant_Comments_Classification.pkl']
```

```
# Predicting the trained final model
comment_model.predict(X)
```

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

```
# loading the final model
model = joblib.load('Malignant_Comments_Classification.pkl')
```

```
# Adding the predicted values to test dataframe
df_test['Predicted_Values']=Predictions
df_test
```

	id	comment_text	comment_length	clean_length	Predicted_Values
0	00001cee341fdb12	yo bitch ja rule succesful ever whats hating s...	367	227	1
1	0000247867823ef7	rfc title fine imo	50	18	0
2	00013b17ad220c46	source zawe ashton lapland	54	26	0
3	00017563c3f7919a	look back source information updated correct f...	205	109	0
4	00017695ad8997eb	anonymously edit article	41	24	0
...	...	...	...	...	...
153159	fffcd0960ee309b5	totally agree stuff nothing long crap	60	37	0
153160	fffd7a9a6eb32c16	throw field home plate get faster throwing cut...	198	107	0
153161	fffda9e8d6fafa9e	okinotorishima category see change agree corre...	423	238	0
153162	fffe8f1340a79fc2	one founding nation eu germany law return quit...	502	319	1
153163	ffffce3fb183ee80	stop already bullshit welcome fool think kind ...	141	74	0

153164 rows × 5 columns

Here I have added new predicted values to test dataframe. Using classification model, we have got the predicted values of malignant comments.

```
# Checking values counts for predicted values
df_test.Predicted_Values.value_counts()
```

```
0    135136
1     18028
Name: Predicted_Values, dtype: int64
```

```
# Saving the data into csv file
df_test.to_csv("MalignantCommentsPredictedTestData.csv",index=False)
```

Finally, saving my test data into csv file

- Key Metrics for success in solving problem under Consideration

To evaluate the performance of each algorithm, several metrics are defined accordingly, and are discussed briefly below.

• **Accuracy score**: This metric measures how many of the comments are labelled correctly. However, in our dataset, where most of comments are not toxic, regardless of performance of model, a high accuracy was achieved. Accuracy is the ratio of number of correct predictions into number of predictions. In binary classification problem, accuracy can be calculated as below,

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN}$$

Where TP = True Positive, TN = True Negative, FP = False Positive, FN = False Negative.

• **Precision and Recall:** Precision and recall were designed to measure the model performance in its ability to correctly classify the malignant comments.
Precision explains what fraction of malignant classified comments are truly malignant, and Recall measures what fraction of malignant comments are labelled correctly.

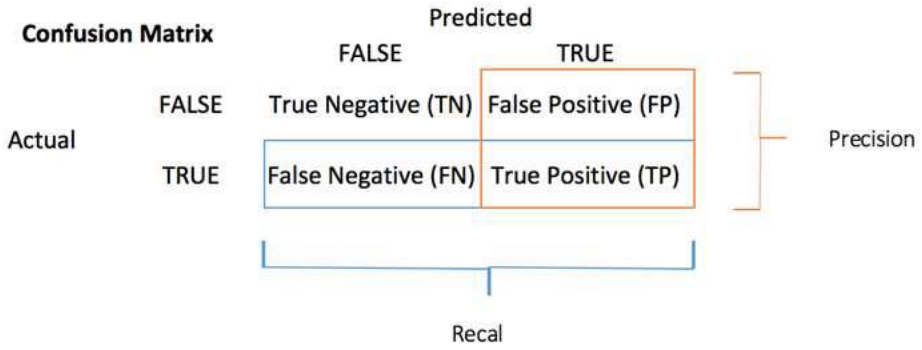
$$\text{Precision} = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

$$\text{Recall} = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

• **F1 Score** is used to express the performance of the machine learning model (or classifier). It gives the combined information about the precision and recall of a model. This means a high F1-score indicates a high value for both recall and precision.

$$F1\ score = 2 * \frac{Precision\ * Recall}{Precision + Recall}$$

• **Confusion Matrix** is one of the evaluation metrics for machine learning classification problems, where a trained model is being evaluated for accuracy and other performance measures. And this matrix is called the confusion matrix since it results in an output that shows how the system is confused between the two classes.



• **Cross Validation Score** is a technique in which we train our model using the subset of the data-set and then evaluate using the complementary

subset of the data-set. It is used to protect against overfitting in a predictive model, particularly in a case where the amount of data may be limited. In cross-validation, you make a fixed number of folds (or partitions) of the data, run the analysis on each fold, and then average the overall error estimate. It is used to estimate the performance of ML models.

• **Roc Auc Score:** The **Receiver Operator Characteristic (ROC)** curve is an evaluation metric for binary classification problems. It is a probability curve that plots the **TPR** against **FPR** at various threshold values.

The **Area Under Curve (AUC)** is the measure of the ability of a classifier to distinguish between classes and is used as a summary of the ROC curve.

• **Log Loss** is the most important classification metric based on probabilities. Log Loss is the negative average of the log of corrected predicted probabilities for each instance. Log loss for binary classification is:

$$\text{Log loss} = \frac{1}{N} \sum_{i=1}^{N} -\left(y_i * \log(p_i) + (1-y_i) * \log(1-p_i) \right)$$

Where pi is the probability of class 1, and (1-pi) is the probability of class 0. Log loss for multi-class classification is:

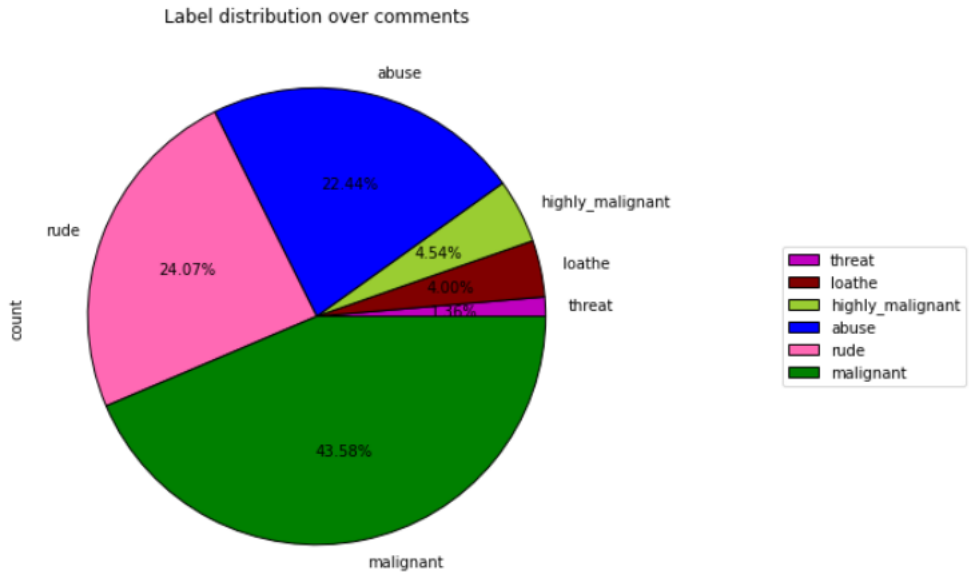
$$logloss = -\frac{1}{N} \sum_{i}^{N} \sum_{j}^{M} y_{ij} \log(p_{ij})$$

N is the number of rows and **M** is the number of classes.

• **Hamming Loss** is the fraction of wrong labels to the total number of labels. In multi-class classification, hamming loss is calculated as the hamming distance between y_true and y_pred. In multi-label classification, hamming loss penalizes only the individual labels.

- Visualizations

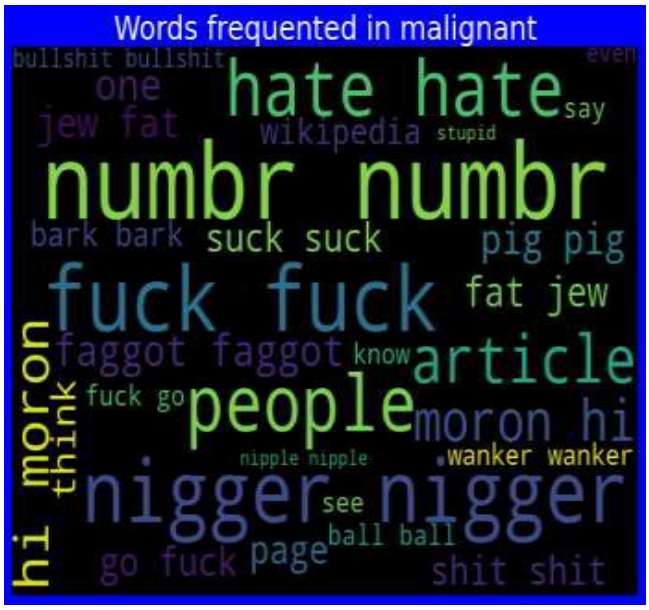
I used pandas profiling to get the over viewed visualization on the pre-processed data. Pandas is an open-source Python module with which we can do an exploratory data analysis to get detailed description of the features and it helps in visualizing and understanding the distribution of each variable. I have used wordcloud to get the sense of loud words in the labels.



Observations:

• From the pie chart we can notice approximately 43% of the comments are malignant, 24% of the comments are rude and 22% are abuse. The count of malignant comment is high compared to other type of comments and the count of threat comments are very less.

Plotting WordCloud for each label:


Words frequented in malignant
bullshit bullshit
even
one
hate hate
say
jew fat
wikipedia
stupid
numbr numbr
bark bark
suck suck
pig pig
fuck fuck
fat jew
faggot faggot
know
article
hi moron
think
fuck go
people
moron hi
nipple nipple
wanker wanker
nigger
see
nigger
go fuck
page
ball ball
shit shit


Words frequented in highly_malignant
cocksucker mothjer
Suck dick
nigger nigger
suck mexican
suck suck
fuck bitch
bitch fuck
yourselfgo fuck
huge faggot
shit shit
die die
shut fuck
fuck yourselfgo
fucker cocksucker
faggot buge
mothjer fucker
go fuck
fuck go
fucksex fucksex
fuck fuck
bastard pro
suck cock
faggot faggot
fuck dog pro assad


Words frequented in rude
jew fat
nipple nipple
article
mothjer fucker
faggot faggot
nigger
think
nigger
bitch fuck
page
die die
suck suck
fat jew
yourselfgo fuck
fucksex fucksex
fuck
fuck
fuck bitch
people
suck cock
buttsecks buttsecks
numbr
numbr
one
go
bullshit bullshit
fuck go
shit shit
suck dick
fuck yourselfgo
dickhead dickhead


Words frequented in threat
kill going
going
murder
pathetic fool
respect supertrnumbrll
wale must
die wikipedia
go
wikipedia die
supertrnumbrll live
must die
di edie
blank talk
supertrnumbrll die
fuck
die pathetic
en wikipedia
die
die
edie die
die di
wikipedia org
talk page
forever respect
lifetime ban
jim wale
fucking
kill kill
live forever

Observations:

• From the above plots we can clearly see the toxic words which are indication of malignant, highly malignant, rude, threat, abuse and loathe words.

• Here most frequent words used for each label is displayed in the word cloud based on different label and when all the values are present.

- Interpretation of the Results

Visualizations: I have used distribution plot to visualize how the data has been distributed. Used count plots and pie charts to check the count of category for each feature. The heat map helped me to understand the correlation between dependent and independent features. Also, heat map helped to detect the multicollinearity problem and feature importance. With the help of WordClouds I would be able to sense the loud words in each label. AUC-ROC curve helped to select the best model.
Pre-processing: The dataset should be cleaned and scaled to build the ML models to get good predictions. I have performed few NLP text processing steps which I have already mentioned in the pre-processing steps where all the important features are present in the dataset and ready for model building.
Model building: After cleaning and processing data, I performed train test split to build the model. I have built multiple classification models to get the accurate accuracy score, and evaluation metrics like precision, recall, confusion matrix, f1 score, log loss, hamming loss. I got Extreme Gradient Boosting Classifier (XGB Classifier) as the best model which gives 94.96% accuracy score. I checked the cross-validation score ensuring there will be no overfitting. After tuning the best model XGB Classifier, I got 95.47% accuracy score and got increment in AUC-ROC curve. Finally, I saved my final model and got the good predictions results for test dataset.

CONCLUSION

- Key Findings and Conclusions of the Study

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.

The conclusion for our study:

o In training dataset, we have only 10% of data which is spreading hate on social media.
o In this 10% data most of the comments are malignant, rude or abuse.
o After using the wordcloud we find that there are so many abusive words present in the negative comments. While in positive comments there is no use of such comments.
o Some of the comments are very long while some are very short

- Learning Outcomes of the Study in respect of Data Science

While working on this project we learned many things and gains new techniques and ways to deal with uncleaned text data. Found how to deal with multiple target features. Tools used for visualizations gives a better understanding of dataset. We have used a lot of algorithms and find that in the classification problem where we have only two labels, XGB Classifier gives better results compared to others.
It is possible to classify the comments content into the required categories of authentic and however, using this kind of project an awareness can be created to know what is fake and authentic.

- Limitations of this work and Scope for Future Work

Limitations: This project was amazing to work on, it creates new ideas to think about but there were some limitations in this project like unbalanced dataset. 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.

Future work: In future work, we can focus on performance and error analysis of the model as lots of comments are misclassified into the hate category. Previous work has achieved success using various algorithms on data in English language but in future, we can consider having data in regional languages. We can also work on after work of the detection of the malignant comments like automatic blocking of the user, auto-deletion of harmful comments on social media platforms. Comment detection is an emerging research area with few public datasets. So, a lot of works need to be done on this field.