Malignant Comment Classifier Project

Submitted by:

Parth Rupavatiya

ACKNOWLEDGMENT

- I would like to express my special thanks to my SME Subham Yadav, who gave me the golden opportunity to do this wonderful project on the topic Malignant Comment Classifier.
- Some of the articles and research papers, I find useful for completion of this project.

REFERENCES:

1) Kevin Khieu, Neha Narwal "Detecting and Classifying Toxic Comments" CS224N.
2) https://towardsdatascience.com/toxic-comment-classification-using-lstm-and-lstm-cnn-db945d6b7986

INTRODUCTION

Business Problem Framing:

- Online forums and social media platforms have provided individuals with the means to put forward their thoughts and freely express their opinion on various issues and incidents. In some cases, these online comments contain explicit language which may hurt the readers.
- Online hate, described as abusive language, aggression, cyber bullying, 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 cyber bullying and trolls on various social media platforms. Many celebrities and influencers 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.
- To protect users from being exposed to offensive language on online forums or social media sites, companies have started flagging

comments and blocking users who are found guilty of using unpleasant language.

- Here, we need to build a model that can classify comments between good comments and toxic comments.

Conceptual Background of the Domain Problem:

- Our aim in this project is to focus mainly on data pre-processing and feature engineering and ensure that the data, which will be consumed by our machine learning models, is as clean as possible.
- In addition, our goal is to build a prototype of online hate and abuse comment classifier which can used to classify good and offensive comments so that it can be controlled and restricted from spreading hatred and cyber bullying.

Review of Literature:

- This project aims to build machine learning model that can classify good and offensive comments. So that offensive comments can be controlled and restricted from spreading hatred and cyber bullying
- First we do all the data pre-processing steps and do EDA to visualize the data graphically and after that we clean and prepare our data using Natural Language Processing (NLP), which is helpful in make a best machine learning model.

Motivation for the Problem Undertaken:

In current scenario, there has been increasing the cases of cyber bullying and trolls on social media platforms. My motivation for this project is to doing some research on this thing and makes a machine learning model. This helps clients to classify offensive comments and remove them from various platforms as well as take some legal action on offensive comments.

Analytical Problem Framing

Mathematical/ Analytical Modelling of the Problem:

- We use some mathematical, statistical and analytics approaches in this project. Which is described below:
    1) **Removing Stop Words**: Stop words are common words that structure a sentence. Words such as “I”, “are”, and “here” do not contribute to the sentiment (the rating in our case) of reviews. Hence, we decided to remove stopwords. We used NLTK’s stopwords package to provide us with the list of stopwords.
    2) **Lemmatization:** Lemmatization is the process of converting a word to its base form. Lemmatization usually refers to doing things properly with the use of a vocabulary and morphological analysis of words, normally aiming to remove inflectional endings only and to return the base or dictionary form of a word, which is known as the lemma. We applied this method on our comments text.
    3) **Term Frequency - Inverse Document Frequency (TF-IDF):** This approach to encoding is also fairly simplistic but carries more information than just the number of occurrences, like Bag-of-Words. TF-IDF is a numerical statistic that also reflects how important a word is to a document or paragraph. The scheme is split into two calculations, TF and IDF.

    - TF, or Term Frequency, is the number of times a term, *t*, appears divided by the number of terms in the document/paragraph, *d*.

$$tf_{t,d} = \frac{n_{t,d}}{Number\ of\ terms\ in\ the\ document}$$

Equation for Term Frequency

    - IDF, or Inverse Document Frequency, is a measure of how important the term is in the corpus. It’s calculated by taking the log of the number of documents divided by the number of documents with the term *t* in it.

$$idf_t = \log \frac{number\ of\ documents}{number\ of\ documents\ with\ term\ 't'}$$

Equation for Inverse Document Frequency

- The TF-IDF encoding scheme outputs TF × IDF for each term in each review.
- We convert our comments text into vectors using TF-IDF method.

4) **Synthetic minority oversampling technique (SMOTE):** In our dataset target variable is imbalanced and it provides inaccurate results during machine learning part. So, we used SMOTE method to cop up with that issue. SMOTE is an oversampling method and one of the most commonly used oversampling method to solve the imbalance classification problem. This method creates synthetic samples (not duplicate) of minority class. These synthetic records are generated by randomly selecting one or more of the k-nearest neighbors for each example in the minority class.

Data Sources and their formats:

- In this project, client provides us two dataset, training dataset and testing dataset. Both dataset are in csv format.
- The dataset contains the training set, which has 1,59,571 samples and the test set which contains 1,53,164 samples. Training dataset contain 8 features which includes 'Id', 'Comments', 'Malignant', 'Highly malignant', 'Rude', 'Threat', 'Abuse' and 'Loathe'. While test dataset contain 2 features which include 'Id' and 'Comments'.
- 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.
- Below is the snapshot of our training dataset:

	id	comment_text	malignant	highly_malignant	rude	threat	abuse	loathe
0	0000997932d777bf	Explanation\nWhy the edits made under my usern...	0	0	0	0	0	0
1	000103f0d9cfb60f	D'aww! He matches this background colour I'm s...	0	0	0	0	0	0
2	000113f07ec002fd	Hey man, I'm really not trying to edit war. It...	0	0	0	0	0	0
3	0001b41b1c6bb37e	"\nMore\nI can't make any real suggestions on ...	0	0	0	0	0	0
4	0001d958c54c6e35	You, sir, are my hero. Any chance you remember...	0	0	0	0	0	0

Data Pre-processing and Data Preparation:

For cleaning and preparation of the data, we use some techniques which are described below:

- We check null values present in the data, but there is no null value found in the dataset.
- There are six target features in this dataset so we combined them and make one target feature giving the name as label and also we do scaling for that feature.
- After data analysis, we clean our data using some NLP (Natural Language Processing) techniques. We convert comments text column in lower case and remove punctuation from that column like +, - , /, etc. because that type of punctuation badly affect our result. Then we remove digits from comments text.
- After that, we apply stop words and lemmatization techniques on comments text, which we already discussed above.
- At last, we use tf-idf and smote techniques for preparation of the data, which we also discussed above.

Data Inputs- Logic- Output Relationships:

- We used classification machine learning models because our target variable is label (0 or 1).
- There are many classification models but here we used some of them models.

- First we split our dataset into two segments: training and testing. We take 80% data for training and 20% data for testing. For splitting data we use train test split method. Below is the code for splitting the data:

```
# split train and test data
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x_smote,y_smote,test_size=0.20,random_state=0)
```

    1) 80% of the observation as training set = x_train
    2) The associated target for each observation in x_train = y_train
    3) 20% of the observation as test set= x_test
    4) The target associated with the test set = y_test.
- After splitting data we passed training data to machine learning models. The fitted model will first be used to generate prediction on the test set (x_test). Next, the predicted class labels are compared to the actual observed class label (y_test).

Libraries Used:

- We used many libraries used in this project, which is described below:
    1) Numpy: This library is used for scientific computing. It supports multidimensional arrays and matrices.
    2) Pandas: This library is used for data analysis and modelling convenient in python. Pandas simplify analysis by converting CSV, JSON, and TSV data files or a SQL database into a data frame with rows and columns.
    3) Matplotlib and Seaborn: Both libraries are used for data visualization.
    4) Imbalanced-learn: This library is used for imbalanced classification of target variable.
    5) Wordcloud: Word Cloud is a data visualization technique used for representing text data in which the size of each word indicates its frequency or importance. Significant textual data points can be highlighted using a word cloud. Word clouds are widely used for analyzing data from social network websites.

6) NLTK (Natural Language Toolkit): It is a suite that contains libraries and programs for statistical language processing. It is one of the most powerful NLP libraries, which contains packages to make machines understand human language and reply to it with an appropriate response.
7) Scikit-learn: The Python library, Scikit-Learn, is built on top of the Matplotlib, Numpy, and SciPy libraries. It has wide range of algorithms.

Models Development and Evaluation

Identification of possible problem-solving approaches:

- Below is the approaches that we used in solve the issue and make the best model:
  1) Data reading and understanding
  2) Data analysis
  3) Data cleaning and preparation
  4) Over sampling
  5) Train test split
  6) Machine learning algorithms

Testing of Identified Approaches (Algorithms):

The classification algorithm that we used is:

1) Logistic Regression
2) Decision tree classifier
3) MultinomialNB
4) Random Forest Classifier
5) Gradient Boosting Classifier
6) Xgboost Classifier
7) Catboost Classifier

Building machine learning models:

- We use many algorithms to find best model, but here we describe only best model.
- We find random forest classifier as a best model. It is supervised learning algorithm. This algorithm generates many individual trees, often hundreds or thousands.
- The "forest" it builds, is an ensemble of decision trees, usually trained with the “bagging” method. The general idea of the bagging method is that a combination of learning models increases the overall result.
- This classifier produces multiple decision trees and merges them together to produce accurate result. Below is the code of our model with random forest classifier:

```
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier()
rf.fit(x_train,y_train)
train = rf.score(x_train,y_train)
pred_rf=rf.predict(x_test)
print("Accuracy of training model is:",train)
print("Accuracy Score:",accuracy_score(y_test,pred_rf))
print("Confusion matrix:","\n",confusion_matrix(y_test,pred_rf))
print("Classification report:","\n",classification_report(y_test,pred_rf))
```

Output:

```
Accuracy of training model is: 0.9968432939617097
Accuracy Score: 0.9756361289872513
Confusion matrix:
 [[27489  1033]
 [  364 28453]]
Classification report:
              precision    recall  f1-score   support

           0       0.99      0.96      0.98     28522
           1       0.96      0.99      0.98     28817

    accuracy                           0.98     57339
   macro avg       0.98      0.98      0.98     57339
weighted avg       0.98      0.98      0.98     57339
```

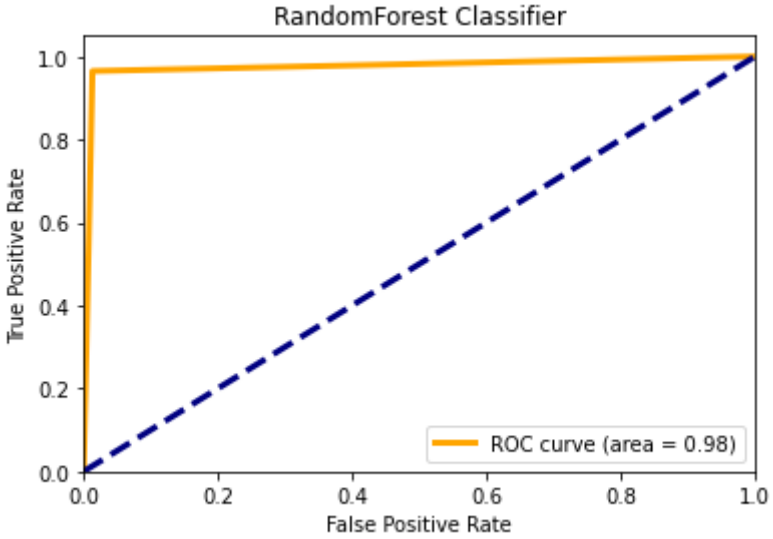
- We get 99% training model accuracy and 97% test data accuracy with best confusion matrix and classification report. As we can see in output, Randomforest classifier also gives good precision and recall score along with f1 score.

- Then, we check auc_roc score and plot auc curve of random forest classifier model. Area Under the 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. The higher the AUC, the better the performance of the model at distinguishing between the positive and negative classes. Below is the code of auc_roc curve:

```
# check auc_roc curve and auc score of best model
from sklearn.metrics import roc_curve,auc
fpr,tpr,thresholds = roc_curve(pred_rf,y_test)
roc_auc = auc(fpr,tpr)

plt.figure()
plt.plot(fpr,tpr,color="orange", lw=3, label=("ROC curve (area = %0.2f)" % roc_auc))
plt.plot([0,1],[0,1],color = "navy",lw=3,linestyle="--")
plt.xlim([0.0,1.0])
plt.ylim([0.0,1.05])
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("RandomForest Classifier")
plt.legend(loc = "lower right")
plt.show()
```

Output:



- From the above plot we can see that, we get best area under the curve for randomforest classifier, which is 98%.

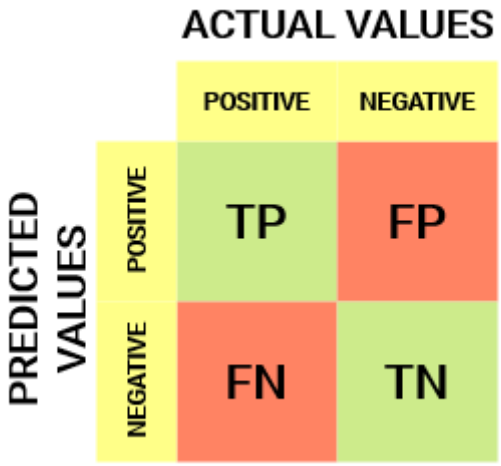
- Now, check accuracy of all used algorithms:

Algorithms	Training accuracy	Testing accuracy	Precision		Recall		f1-score	
			label 0	label 1	label 0	label 1	label 0	label 1
Logistic Regression	0.93	0.92	0.94	0.92	0.92	0.94	0.93	0.93
Decision tree classifier	0.99	0.94	0.96	0.93	0.93	0.96	0.94	0.95
MultinomialNB	0.89	0.89	0.88	0.9	0.91	0.88	0.89	0.89
Random Forest Classifier	0.99	0.97	0.99	0.96	0.96	0.99	0.98	0.98
Gradient Boosting Classifier	0.83	0.83	0.76	0.95	0.96	0.7	0.85	0.81
Catboost Classifier	0.95	0.93	0.93	0.94	0.94	0.93	0.94	0.94
Xgboost Classifier	0.91	0.9	0.86	0.96	0.97	0.85	0.91	0.9

Key Metrics for success in solving problem under Consideration:

- We use three types of metrics for solving problem. Which is described below:
    1) Accuracy score: Accuracy score is a metric for evaluating classification model. It is calculated by number of correct prediction made divided by the total number of prediction made.

    2) Confusion matrix: A confusion matrix is an nxn matrix used for evaluating performance of classification model, where n is the number of target classes. The matrix compares the target values with predicted values which are predicted by machine learning model. This matrix gives us a view of how good our classification model work and what kind of errors t is making. Below is the simple figure of confusion matrix.



- If the model predicts that a client will enter default, and client really enters into default, then it is true positive situation (TP).
- When the model classifies a case as positive, but it actually negative, then it is a false positive observation (FP).
- If the model predicts that a client will not enter into default, and the client didn't entered into default in the test set, then it is true negative situation (TN).
- The prediction of the model may be negative, while the client may actually enter into default, then it is false negative classification (FN).
- Below is the output of our best confusion matrix with random forest classifier:

```
[[27489  1033]
[  364 28453]]
```

3) Classification Report: A classification report is used to measure the quality of predictions from a classification algorithm. This report shows the metrics like precision, recall and f1- score. The metrics are calculated by using true and false positives, true and false negatives.
   - Precision depicts how many of the correctly predicted cases actually turned out to be positive. Below is the formula of precision and calculation of precision with random forest classification model :

$$Precision = \frac{TP}{TP + FP}$$

Precision = 27489/ (27489+1033)
Precision = 0.96

- Recalls depicts how many of actual positive cases we were able to predict correctly with our model. Below is the formula of recall:

$$Recall = \frac{TP}{TP + FN}$$

Recall = 27489/ (27489+364)
Recall = 0.99

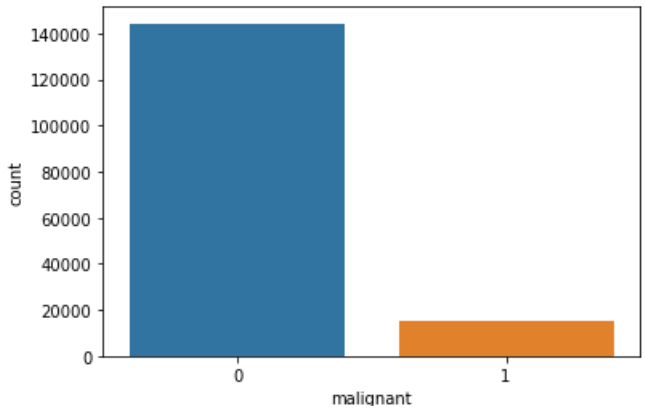
- F1- score gives combined idea about precision and recall metrics. Below is the formula of f1-score:

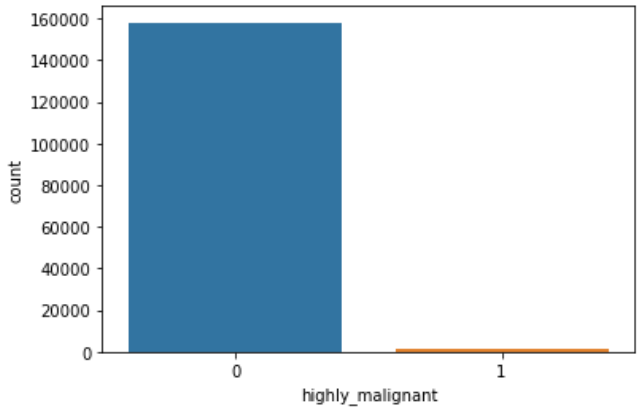
$$F1 - score = \frac{2}{\frac{1}{Recall} + \frac{1}{Precision}}$$

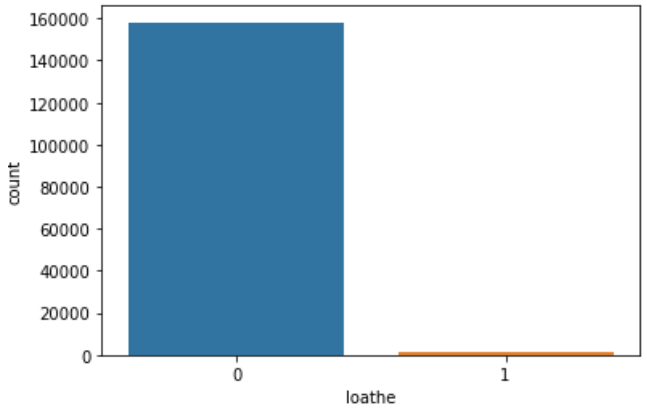
F1 –score = 2/ ((1/0.99) + (1/0.96))
F1- score = 0.98

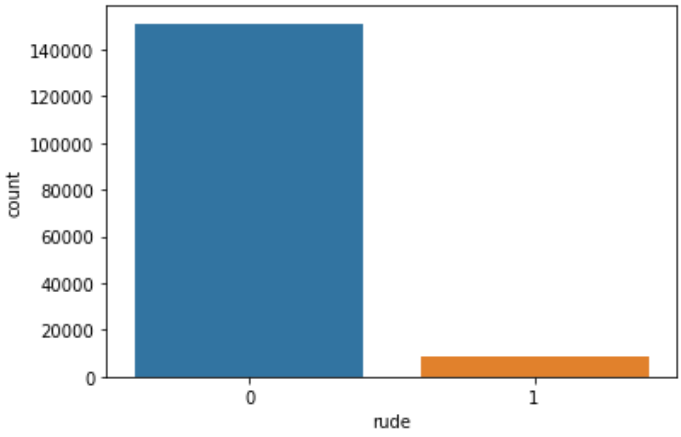
Visualizations:

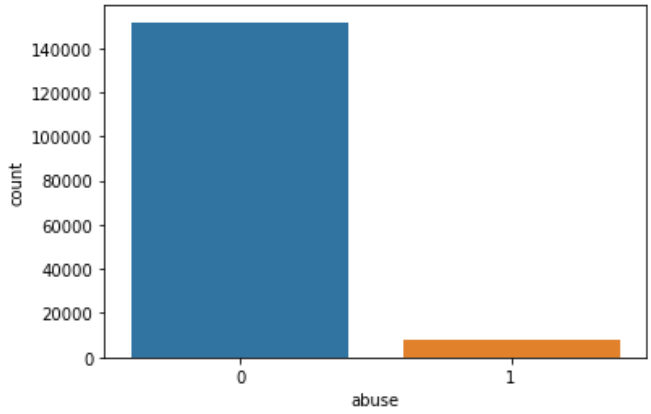
- Now, let's look at the analysis of the data.
- First, we plot the target feature columns.

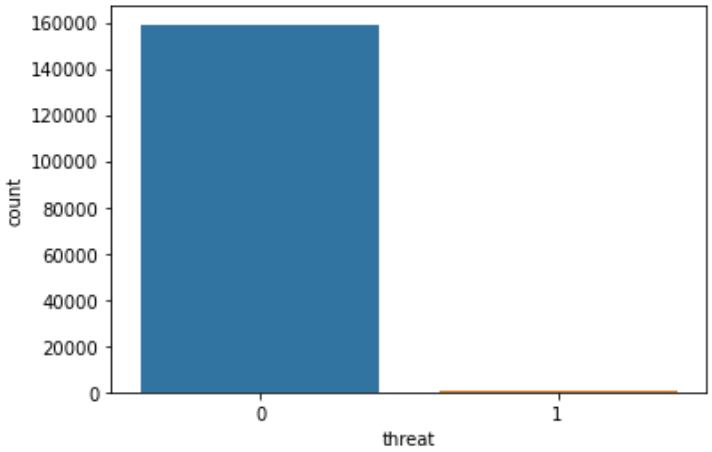



160000
140000
120000
100000
80000
60000
40000
20000
0
count
0
1
highly_malignant

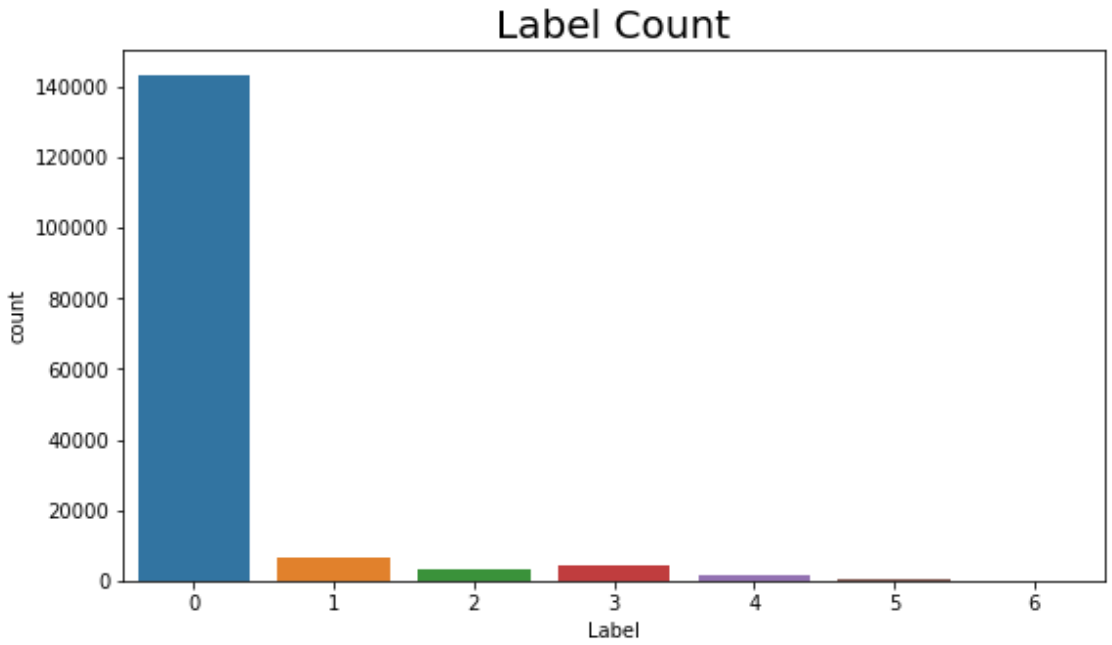

160000
140000
120000
100000
80000
60000
40000
20000
0
count
0
1
loathe


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

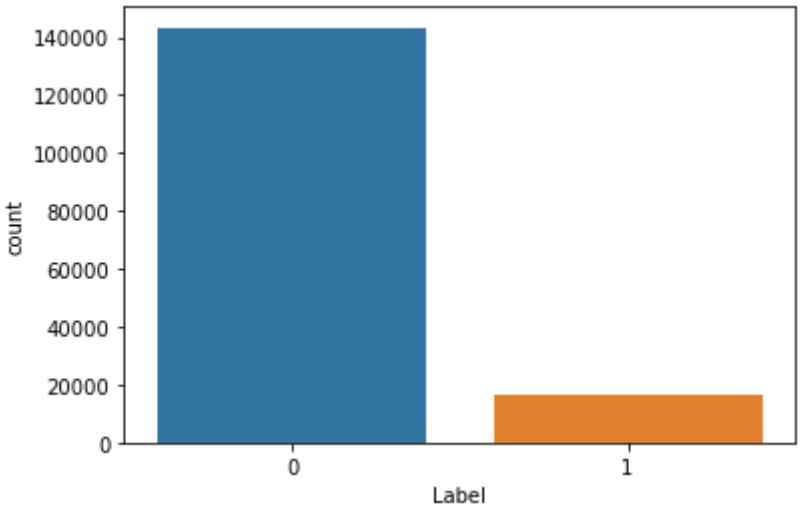

140000
120000
100000
80000
60000
40000
20000
0
count
0
1
abuse



Above all count plot is our target feature and we can see that there is imbalanced classification and this issue affect our final result. For solve this issue we use oversampling method, which we already discussed above.



From the above plot we can say that, most of the comments are good and very less numbers of comments are toxic. 0 depicts good comments, while 1, 2, 3, 4, 5 and 6 depicts offensive comments.



From the above count plot we can see that, after merge all target columns, it is still imbalanced. So, to solve this issue we used oversampling (SMOTE) method.

Above Wordcloud is for malignant comments in label column. This Wordcloud is highlight popular words and phrases based on frequency and relevance.

Above Wordcloud is for abuse comments in label column. This Wordcloud is highlight popular words and phrases based on frequency and relevance.

CONCLUSION

Key Findings and Conclusions of the Study:

- The purpose of this article was twofold: to understand the pattern of various types of comments and make predictive model able to effectively classify observation in the two classes, i.e. 0 (good comments) and 1 (offensive comments)
- Our target variable is imbalanced, so we use SMOTE technique to get a good result.
- We use many classifiers to find best model and best result were observed for the random forest classifier with 97% accuracy score. We also get a good precision and recall score along with f1-score.
- Our model is able to detect offensive comments with very high confidence.

Limitation of this work and Scope for Future Work:

- I used i3 processor computer for making this model, so I am not able to use GridSearchCV and cross validation because system takes too much time to give the result and sometimes it also gives memory error. If somehow I use GridSearchCV and Cross validation then our model result will be surely improved.
- For future work, the model should be further improved by trying to make deep learning model. Moreover, try other sampling technique to get best results.