PROJECT REPORT ON:

"Malignant Comment Classifier"

SUBMITTED BY

Sumit Dhandhania

ACKNOWLEDGMENT

I would like to express my special gratitude to “Flip Robo” team, who has given me this opportunity to deal with a beautiful dataset and it has helped me to improve my analyzation skills. Mr. Shwetank Mishra (SME Flip Robo) who has inspired me in so many aspects and also encouraged me a lot with his valuable words and with his unconditional supportI have ended up with a beautiful Project.

A huge thanks to my academic team “Data trained” who are the reason behind what I am today. Last but not least my parents who have been my backbone in every step of my life. And also thank you for many other persons who provided me with assistance directly or indirectly to complete the project.

Contents:

1.INTRODUCTION

1.1 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.

1.2 Conceptual Background of the Domain Problem

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 cyber bullying.

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

1.3 Review of Literature

Nowadays users leave numerous comments on different social networks, news portals, and forums. Some of the comments are toxic or abusive. Due to numbers of comments, it is unfeasible to manually moderate them, so most of the systems use some kind of automatic discovery of toxicity using machine learning models. In this work, we performed a systematic review of the state-of-the-art in toxic comment classification using machine learning methods. First, we have investigated when and where the papers were published and their maturity level. In our analysis of every primary study we investigated: data set used, evaluation metric, used machine learning methods, classes of toxicity, and comment language.

1.4 Motivation for the Problem Undertaken

The project was the first provided to me by FlipRobo as a part of the internship program. The exposure to real world data and the opportunity to deploy my skillset in solving a real time problem has been the primary objective. However,

the motivation for taking this project was that it is relatively a new field of research. Here we have many options but less concrete solutions. The main motivation is to build a prototype of online hate and abuse comment classifier which can used to classify hate and offensive comments so that it can be controlled and restricted from spreading hatred and cyberbullying.

2.Analytical Problem Framing

2.1 Mathematical/ Analytical Modeling of the Problem

In this perticular problem the label can be either 0 or 1, where 0 denotes a NO while 1 denotes a YES. So clearly it is a binary classification problem and I have to use all classification algorithms while building the model. We would perform one type of supervised learning algorithms: Classification. While it seems more reasonable to perform Classification since we have 5-6 class to predict. Here, we will only perform classification. Since there only 1 feature in the dataset, filtering the words is needed to prevent overfit. In order to determine the regularization parameter, throughout the project in classification part, we would first remove email, phone number, web address, spaces and stops words etc. In order to further improve our models, we also performed TFID in order to convert the tokens from the train documents into vectors so that machine can do further processing. I have used all the classification algorithms while building model then tunned the best model and saved the best model. At last I have predicted the Malignance using saved model.

2.2 Data Sources and their formats

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.

This project is more about exploration, feature engineering and classification that can be done on this data. Since the data set is huge and includes many categories of comments, we can do good amount of data exploration and derive some interesting features using the comments text column available.

We need to build a model that can differentiate between comments and its cat egories.

2.3 Data Preprocessing Done

- ✓ As a first step I have imported required libraries and I have imported the dataset which was in csv format.
- ✓ Cleaned the data from junk values. Replace multiple spaces with single space So that it will be easy to classify it.
- ✓ I am creating a function for feature engineering and making three different columns using comment_text column Length: indicating the length of the text. Exclamation: indicates whether '!' is present in the text or not. Question: indicates whether '?' is present in the text or not.
- ✓ By observing these comments we can say that we need to do lot of text processing as there are many words which are not important for prediction, as well as numbers and other stuff.

2.4 Hardware and Software Requirements and Tools Used

While taking up the project we should be familiar with the Hardware and software required for the successful completion of the project. Here we need the following hardware and software.

Hardware required: -

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

Software/s required: -

1.Anaconda

Libraries required :-

- ✓ To run the program and to build the model we need some basic libraries as follows:

```
In [1]: #importing required libraries
        import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        import seaborn as sns
        import datetime as dt

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

- ✓ **import pandas as pd**: **pandas** is a popular Python-based data analysis toolkit which can be imported using import pandas as pd. It presents a diverse range of utilities, ranging from parsing multiple file formats to converting an entire data table into a numpy matrix array. This makes pandas a trusted ally in data science and machine learning.
- ✓ **import numpy as np**: NumPy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting,

selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more.

- ✓ **import seaborn as sns:** Seaborn is a data visualization library built on top of matplotlib and closely integrated with pandas data structures in Python. Visualization is the central part of Seaborn which helps in exploration and understanding of data.
- ✓ **Import matplotlib.pyplot as plt:** matplotlib.pyplot is a collection of functions that make matplotlib work like MATLAB. Each pyplot function makes some change to a figure: e.g., creates a figure, creates a plotting area in a figure, plots some lines in a plotting area, decorates the plot with labels, etc.
- ✓ from sklearn.svm import LinearSVC
- ✓ from sklearn.naive_bayes import MultinomialNB
- ✓ from sklearn.linear_model import LogisticRegression
- ✓ from lightgbm import LGBMClassifier
- ✓ from sklearn.linear_model import SGDClassifier
- ✓ from sklearn.metrics import classification_report
- ✓ from sklearn.metrics import accuracy_score
- ✓ from sklearn.model_selection import cross_val_score

With this sufficient libraries we can go ahead with our model building.

3.Data Analysis and Visualization

3.1 Identification of possible problem-solving approaches (methods)

Just make the comments more appropriate so that we'll get less word to process and get more accuracy. Removed extra spaces, converted email address into email keyword, likely wise phone number etc. Tried to make Comments small and more appropriate as much as it was possible.

3.2 Testing of Identified Approaches (Algorithms)

In this nlp based project we need to predict multiple targets which are binary. I have converted the text into vectors using TFIDF vectorizer and separated our feature and labels then build the model using One Vs Rest Classifier. Among all the algorithms which I have used for this purpose I have chosen LinearSVC as best suitable algorithm for our final model as it is performing well compared to other algorithms while evaluating with different metrics I have used following algorithms and evaluated them

- LinearSVC
- LogisticRegression
- MultinomialNB
- LightGBMClassifier
- SGDClassifier

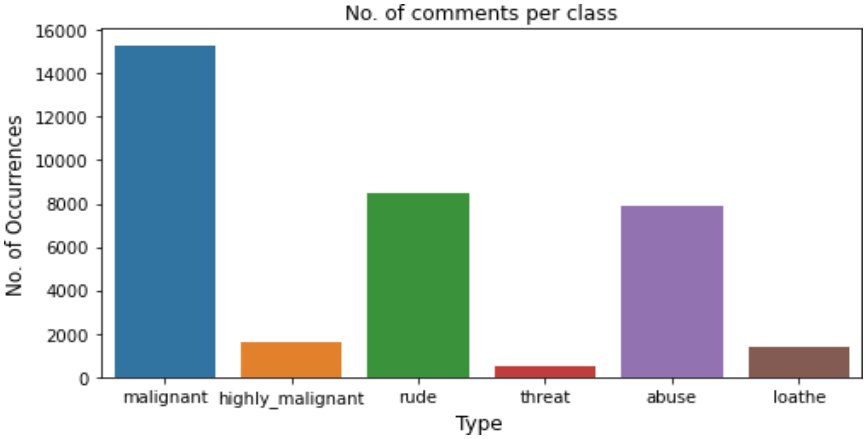
From all of these above models LinearSVC was giving me good performance.

3.3 Key Metrics for success in solving problem under consideration

I have used the following metrics for evaluation:

- I have used f1_score, precision_score, recall_score, multilabel_confusion_matrix and hamming loss all these evaluation metrics to select best suitable algorithm for our final model.
- **Precision** can be seen as a measure of quality, higher precision means that an algorithm returns more relevant results than irrelevant ones.
- **Recall** is used as a measure of quantity and high recall means that an algorithm returns most of the relevant results.
- **Accuracy score** is used when the True Positives and True negatives are more important. Accuracy can be used when the class distribution is similar.
- **F1-score** is used when the False Negatives and False Positives are crucial. While F1-score is a better metric when there are imbalanced classes.

3.4 Visualizations



✓ The above figure represents count plot for all our labels. Looking at this plot we can conclude that more number of comments has been labelled as malignant compared to others. Very less number of comments has been labelled as threat.

Words frequented in malignant

Words frequented in highly_malignant

✓ The above both figures are representing the word occurrence in case of malignant and highly malignant comments respectively.

Words frequented in threat

Words frequented in rude

✓ The above both figures are representing the word occurrence in case of threat and highly rude comments respectively.

3.5 Run and Evaluate selected models

1. Model Building:

In [41]:
```
#Lets define different algorithms
svc = LinearSVC()
lr = LogisticRegression(solver='lbfgs')
mnb = MultinomialNB()
lgb = LGBMClassifier()
sgd = SGDClassifier()
```

In [42]:
```
#function for printing score
def print_score(y_pred,clf):
    print('classifier:',clf.__class__.__name__)
    print("Jaccard score: {}".format(jaccard_score(y_test,y_pred,average='micro')))
    print("Accuracy score: {}".format(accuracy_score(y_test,y_pred)))
    print("f1_score: {}".format(f1_score(y_test,y_pred,average='micro')))
    print("Precision : ", precision_score(y_test,y_pred,average='micro'))
    print("Recall: {}".format(recall_score(y_test,y_pred,average='micro')))
    print("Hamming loss: ", hamming_loss(y_test,y_pred))
    print("Confusion matrix:\n ", multilabel_confusion_matrix(y_test,y_pred))
    print('=======================================\n')
```

In [43]:
```
#models with evaluation using OneVsRestClassifier
for classifier in [svc,lr,mnb,sgd,lgb]:
   clf = OneVsRestClassifier(classifier)
   clf.fit(x_train,y_train)
   y_pred = clf.predict(x_test)
   print_score(y_pred, classifier)
```

```
classifier: LinearSVC
Jaccard score: 0.545426673479816
Accuracy score: 0.9194846213621437
f1_score: 0.7058590133580215
Precision :  0.8491646778042959
Recall: 0.6039379880049791
Hamming loss: 0.018583042973286876
Confusion matrix:
  [[[35695   383]
  [ 1267  2548]]

 [[39421    66]
  [  306   100]]

 [[37589   161]
  [  694  1449]]

 [[39769    19]
  [   82    23]]
```

```
 [[37610   272]
  [  873  1138]]

 [[39489    47]
  [  278    79]]]
=======================================

classifier: LogisticRegression
Jaccard score: 0.5223303001778057
Accuracy score: 0.9198355601233299
f1_score: 0.6862246650635521
Precision :  0.8733823015040224
Recall: 0.565123910829467
Hamming loss: 0.01908020621830062
Confusion matrix:
  [[[35821   257]
  [ 1430  2385]]

 [[39416    71]
  [  290   116]]

 [[37620   130]
  [  774  1369]]

 [[39781     7]
  [   92    13]]

 [[37650   232]
  [  960  1051]]

 [[39509    27]
  [  297    60]]]
=======================================

classifier: MultinomialNB
Jaccard score: 0.43909710391822826
Accuracy score: 0.9130925224976812
f1_score: 0.6102397158922758
Precision :  0.8813849113058346
Recall: 0.4666742107049904
Hamming loss: 0.022008873737247137
Confusion matrix:
  [[[35901   177]
  [ 1827  1988]]

 [[39440    47]
  [  328    78]]

 [[37624   126]
  [ 1002  1141]]

 [[39788     0]
  [  105     0]]

 [[37696   186]
  [ 1123   888]]

 [[39517    19]
```

```
  [  328    29]]]
```

```
classifier: SGDClassifier
Jaccard score: 0.44780456306500216
Accuracy score: 0.9143709422705738
f1_score: 0.618598082212146
Precision :  0.9014298093587522
Recall: 0.470861151974652
Hamming loss: 0.02143650932912207
Confusion matrix:
  [[[35976   102]
  [ 1897  1918]]

 [[39487     0]
  [  406     0]]

 [[37619   131]
  [  835  1308]]

 [[39788     0]
  [  105     0]]

 [[37664   218]
  [ 1099   912]]

 [[39532     4]
  [  334    23]]]
========================================

classifier: LGBMClassifier
Jaccard score: 0.5359082679541339
Accuracy score: 0.9174291229037675
f1_score: 0.6978388998035363
Precision :  0.8282294419399969
Recall: 0.6029195428312776
Hamming loss: 0.019276564810869076
Confusion matrix:
  [[[35781   297]
  [ 1412  2403]]

 [[39390    97]
  [  308    98]]

 [[37534   216]
  [  617  1526]]

 [[39735    53]
  [   82    23]]

 [[37498   384]
  [  828  1183]]

 [[39478    58]
  [  262    95]]]
```

==========

2. Hyper Parameter Tunning:

I have did hyperparameter tuning for LinearSVC for the parameters like 'estimator__penalty, 'estimator__loss', 'estimator__multi_class', ' estimator__dual', 'estimator__intercept_scaling', 'estimator__C'.

```
In [44]: param = {
             'estimator__penalty': ['l1'],
             'estimator__loss': ['hinge','squared_hinge'],
             'estimator__multi_class': ['ovr','crammer_singer'],
             'estimator__dual': [False],
             'estimator__intercept_scaling': [2,4,5],
             'estimator__C': [2]
             }
```

```
In [45]: #train the model with given parameters using GridSearchCV
         svc = OneVsRestClassifier(LinearSVC())
         GCV =  GridSearchCV(svc,param,cv = 3, verbose =0,n_jobs=-1)
         GCV.fit(x_train,y_train)
```

```
Out[45]: GridSearchCV(cv=3, estimator=OneVsRestClassifier(estimator=LinearSVC()),
                      n_jobs=-1,
                      param_grid={'estimator__C': [2], 'estimator__dual': [False],
                                  'estimator__intercept_scaling': [2, 4, 5],
                                  'estimator__loss': ['hinge', 'squared_hinge'],
                                  'estimator__multi_class': ['ovr', 'crammer_singer'],
                                  'estimator__penalty': ['l1']})
```

```
In [46]: #printing the best parameters found by GridSearchCV
         GCV.best_params_
```

```
Out[46]: {'estimator__C': 2,
          'estimator__dual': False,
          'estimator__intercept_scaling': 2,
          'estimator__loss': 'squared_hinge',
          'estimator__multi_class': 'ovr',
          'estimator__penalty': 'l1'}
```

- ✓ And after doing hyperparameter tuning I got above parameters as best suitable parameters for our final model.
- ✓ I have tested my final model using these parameters and got better results compared to earlier results for my final model.

```
In [47]: model = OneVsRestClassifier(LinearSVC(C=2,dual = False, loss='squared_hinge',multi_class='ovr', penalty ='l1',intercept_scaling=4
         model.fit(x_train,y_train)
         y_pred = model.predict(x_test)

         print("Jaccard score: {}".format(jaccard_score(y_test,y_pred,average='micro')))
         print("Accuracy score: {}".format(accuracy_score(y_test,y_pred)))
         print("f1_score: {}".format(f1_score(y_test,y_pred,average='micro')))
         print("Precision : ", precision_score(y_test,y_pred,average='micro'))
         print("Recall: {}".format(recall_score(y_test,y_pred,average='micro')))
         print("Hamming loss: ", hamming_loss(y_test,y_pred))
         print("\nConfusion matrix: \n", multilabel_confusion_matrix(y_test,y_pred))
```

```
Jaccard score: 0.5499033472377658
Accuracy score: 0.919559822525255
f1_score: 0.7095969541814362
Precision :  0.8449273096764108
Recall: 0.611632907095168
Hamming loss:  0.018482774755805113

Confusion matrix:
 [[[35680   398]
  [ 1239  2576]]

 [[39413    74]
  [  296   110]]

 [[37581   169]
  [  680  1463]]

 [[39762    26]
  [   76    29]]

 [[37606   276]
  [  874  1137]]

 [[39487    49]
  [  267    90]]]
```

- ✓ After training and building our final model I used this model to make predictions for test dataset. Before doing predictions the test dataset has been cleaned and processed with the same functions which are used for train dataset. And then doing vectorization I have predicted the output labels with our final model.

3. Saving the model and Predictions:

- I have saved my best model using .pkl as follows.

```
In [49]: import joblib
         joblib.dump(model,"Malignant_comment.pkl")

Out[49]: ['Malignant_comment.pkl']

In [50]: #Loading the model
         model = joblib.load('Malignant_comment.pkl')
```

- Now loading my saved model and predicting the values for test dataset.

```
In [58]: pred=pd.DataFrame(predictions, columns = ['malignant','highly_malignant','rude','threat','abuse','loathe'])
         pred
```

Out[58]:

	malignant	highly_malignant	rude	threat	abuse	loathe
0	0	0	0	0	0	0
1	0	0	0	0	0	0
2	0	0	0	0	0	0
3	0	0	0	0	0	0
4	0	0	0	0	0	0
...	...	...	...	...	...	...
153159	0	0	0	0	0	0
153160	0	0	0	0	0	0
153161	0	0	0	0	0	0
153162	0	0	0	0	0	0
153163	0	0	0	0	0	0

153164 rows × 6 columns

4.CONCLUSION

4.1 Key Findings and Conclusions of the Study

For this project we have provided with huge amount of comments with multiple targets which are binary in nature. I observe that there are many words with incorrect spellings. At first I have created three columns one is with the length of the text, another as ‘question’ whether the comment contains ‘?’ mark or not and third as ‘exclamation’ whether the comment contains ‘!’ mark. To clean the column comment_text I have gone through different text processing steps like lowercasing the text, removing unwanted elements like stopwords, ‘\n’, Urls, numbers, punctuations etc. As the text column is with many miss-spelled words and the problem is multi-labelled so we are getting slightly lower accuracy for this task. However we have selected best model among all the algorithms. There are some comments which are from different language other than English we can try the same approach by removing those comments with other languages.

4.2 Learning Outcomes of the Study in respect of Data Science

I found that the dataset was quite interesting to handle. Improvement in computing technology has made it possible to examine social information that cannot previously be captured, processed and analyzed. New analytical techniques of machine learning can be used in property research. The power of visualization has helped us in understanding the data by graphical representation it has made me to understand what data is trying to say. Data cleaning is one of the most important steps to remove unrealistic values and stopwords. This study is an exploratory attempt to use four machine learning algorithms in estimating malignant comments, and then compare their results.

To conclude, the application of machine learning in malignant classification is still at an early stage. We hope this study has moved a small step ahead in providing some methodological and empirical contributions to crediting institutes, and presenting an alternative approach to the valuation of malignance.

4.3 Limitations of this work and Scope for Future Work

Additionally, the followings are some suggested studies to be considered as future work in this area:

a) We suggest a plan to improve the NLP classifiers: first by using other algorithms which such as Support Vector Clustering (SVC) and Convolutional Neural Networks (CNN); secondly, extend the classifiers to the overall goal which is multi-label classifiers. in the current study, the problem simplified into two classes but it worth to pursue a main goal which is 6 classes of comments.

b) We also suggest using SVM for text processing and text classification. It requires a grid search for hyper-parameter tuning to get the best results.