MALIGNANT COMMENTS CLASSIFIER

Submitted by:

ASHOK KUMAR SHARMA

ACKNOWLEDGMENT

I would like to express my deep sense of gratitude to my SME (Subject Matter Expert) **Mr. Shubham Yadav** as well as **Flip Robo Technologies** who gave me the golden opportunity to project on **Malignant Comments Classifier**, which also helped me in doing lots of research and I came to know about so many new things.

I am very much thankful to **Mr. Shankargouda Tegginmani, Trainer (DataTrained)**, for their valuable guidance, keen interest and encouragement at various stages of my training period which eventually helped me a lot in doing this project.

I also acknowledge with thanks for suggestion and timely guidance, which I have received from my SME Mr. Shubham Yadav during this project, which immensely helped me in the evaluation of my ideas on the project.

ASHOK KUMAR SHARMA

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.

- ## 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 cyberbullying.

- ## Review of Literature

  1. What is Comment?

     A comment is a remark or observation that expresses a person's observation or criticism.

- ## Motivation for the Problem Undertaken

  This model can be used by the social media websites where user expresses their opinion by commenting text. This model can classify hate and offensive comments so that it can be controlled and restricted from spreading hatred and cyberbullying.

Analytical Problem Framing

- Mathematical/ Analytical Modeling of the Problem

  For checking datatypes and null values, pandas.DataFrame.info() and pandas.Series.isnull().sum() method has been used. To drop the null values pandas.DataFrame.dropna() method has been used. To replace and remove the certain terms and punctuations, pandas.Series.str.replace() method with regular expression has been used. To get rid of stop words, nltk.corpus.stopwords() method has been used. For stemming words, nltk.stem.SnowballStemmer() has been used.

- Data Sources and their formats

  The data set contains the training set, which has approximately 1,59,571 samples and the test set which contains nearly 1,53,000 samples. All the data samples contain 8 fields which includes 'Id', 'Comments', 'Malignant', 'Highly malignant', 'Rude', 'Threat', 'Abuse' and 'Loathe'. The label can be either 0 or 1, where 0 denotes a NO while 1 denotes a YES. There are various comments which have multiple labels. The first attribute is a unique ID associated with each comment.

  The data set includes:

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

- Data Pre-processing Done

  The following pre-processing pipeline is required to perform model prediction:

1. Load dataset
2. Remove null values
3. Drop column id
4. Convert comment text to lower case and replace '\n' with single space.
5. Keep only text data ie. a-z' and remove other data from comment text.
6. Remove stop words and punctuations
7. Apply Stemming using SnowballStemmer
8. Covert text to vectors using TfidfVectorizer
9. Load saved or serialized model
10. Predict values

- Data Inputs- Logic- Output Relationships

Input	Logic (algorithm)	Output
`comment_text (object)`	GaussianNB MultinomialNB	[malignant, highly_malignant, rude, threat, abuse, loathe]

There is 1 input variable needs to be provided to the logic to get the output i.e. [malignant, highly_malignant, rude, threat, abuse, loathe]. Logic highlighted in green i.e. MultinomialNB is the best performing algorithm among all other logics on this dataset.

- Hardware and Software Requirements and Tools Used

During this project, following set of hardware is being used:

RAM: 8 GB

CPU: AMD A8 Quad Core 2.2 Ghz

GPU: AMD Redon R5 Graphics

and the following software and tools is being used:

a. Python
b. Jupyter Notebook
c. Anaconda

With following libraries and packages:

- Pandas

- Numpy
- Matplotlib
- Seaborn
- nltk
- wordcloud
- sys
- tqdm.notebook
- timeit
- sklearn
- skmulitlearn

Model/s Development and Evaluation

- Identification of possible problem-solving approaches (methods)

  To solve this problem following steps are used:

  1. Load Dataset using pandas
  2. Remove null values
  3. Drop column id
  4. Convert comment text to lower case and replace '\n' with single space.
  5. Keep only text data ie. a-z' and remove other data from comment text.
  6. Remove stop words and punctuations
  7. Apply Stemming using SnowballStemmer
  8. Covert text to vectors using TfidfVectorizer
  9. Separate Input and Output Variables.
  10. Train & Test the Model by supplying Input and Output Variables.

- Testing of Identified Approaches (Algorithms)

  Following are the list of algorithms used for training and testing:

  1. GaussianNB
  2. MultinomialNB

- Run and Evaluate selected models

  A total of 2 algorithm has been used on this dataset for training testing purpose, these are GaussianNB and MultinomialNB. To perform training and testing operation(s) following functions has been defined for which codes are as follows:

```
#importing required libraries

from skmultilearn.problem_transform import BinaryRelevance

from sklearn.naive_bayes import MultinomialNB, GaussianNB

from sklearn.ensemble import AdaBoostClassifier, BaggingClassifier

from sklearn.metrics import hamming_loss, log_loss, accuracy_score,
classification_report

import timeit, sys

import tqdm.notebook as tqdm

#Function to train and test model

def build_models(models,x,y,test_size=0.33,random_state=42):

    #spliting train test data using train_test_split

    x_train,x_test,y_train,y_test =
train_test_split(x,y,test_size=test_size,random_state=random_state)


    #training models using BinaryRelevance of problem transform

    for i in tqdm.tqdm(models,desc="Building Models"):

        start_time = timeit.default_timer()



sys.stdout.write("\n=================================================================
==============================\n")

        sys.stdout.write(f"Current Model in Progress: {i} ")


sys.stdout.write("\n=================================================================
==============================\n")
```

```
        br_clf =
BinaryRelevance(classifier=models[i]["name"],require_dense=[True,True])
        print("Training: ",br_clf)
        br_clf.fit(x_train,y_train)

        print("Testing: ")
        predict_y = br_clf.predict(x_test)

        ham_loss = hamming_loss(y_test,predict_y)
        sys.stdout.write(f"\n\tHamming Loss  : {ham_loss}")

        ac_score = accuracy_score(y_test,predict_y)
        sys.stdout.write(f"\n\tAccuracy Score: {ac_score}")

        cl_report = classification_report(y_test,predict_y)
        sys.stdout.write(f"\n{cl_report}")

        end_time = timeit.default_timer()
        sys.stdout.write(f"Completed in [{end_time-start_time} sec.]")

        models[i]["trained"] = br_clf
        models[i]["hamming_loss"] = ham_loss
        models[i]["accuracy_score"] = ac_score
        models[i]["classification_report"] = cl_report
        models[i]["predict_y"] = predict_y
        models[i]["time_taken"] = end_time - start_time


sys.stdout.write("\n=============================================================
==============================\n\n\n")

    models["x_train"] = x_train
    models["y_train"] = y_train
    models["x_test"] = x_test
    models["y_test"] = y_test
```

```
    return models


#### preparing list of models
models = {
    "GaussianNB": {
        "name":GaussianNB(),
    },
    "MultinomialNB":{
        "name":MultinomialNB(),
    },
}



#taking the one forth of the data for training and testig
half = len(df)//4
trained_models = build_models(models,X[:half,:],Y[:half,:])
```

Building Models: 100%
2/2 [01:21<00:00, 36.01s/it]

```
================================================================================
============
Current Model in Progress: GaussianNB
================================================================================
============
Training:  BinaryRelevance(classifier=GaussianNB(), require_dense=[True, Tr
ue])
Testing:

        Hamming Loss  : 0.21560957083175086
        Accuracy Score: 0.4729965818458033
              precision    recall  f1-score   support

           0       0.16      0.79      0.26      1281
           1       0.08      0.46      0.13       150
           2       0.11      0.71      0.19       724
           3       0.02      0.25      0.03        44
           4       0.10      0.65      0.17       650
           5       0.04      0.46      0.07       109
```

```
   micro avg       0.11      0.70      0.20      2958
   macro avg       0.08      0.55      0.14      2958
weighted avg       0.12      0.70      0.21      2958
 samples avg       0.05      0.07      0.05      2958
Completed in [66.71356009999994 sec.]
================================================================================
============
```

```
================================================================================
============
Current Model in Progress: MultinomialNB
================================================================================
============
Training:  BinaryRelevance(classifier=MultinomialNB(), require_dense=[True,
True])
Testing:

        Hamming Loss  : 0.024091657171793898
        Accuracy Score: 0.9074060007595898
              precision    recall  f1-score   support

           0       0.94      0.48      0.63      1281
           1       1.00      0.01      0.01       150
           2       0.93      0.45      0.60       724
           3       0.00      0.00      0.00        44
           4       0.84      0.35      0.49       650
           5       0.00      0.00      0.00       109

   micro avg       0.91      0.39      0.55      2958
   macro avg       0.62      0.21      0.29      2958
weighted avg       0.87      0.39      0.53      2958
 samples avg       0.04      0.03      0.04      2958
Completed in [14.287291100000061 sec.]
================================================================================
============
```

From the above model comparison it is clear that **MultinomialNB** performs better with **Accuracy Score: 90.74%** and **Hamming Loss: 2.4%** than other models. Therefore, proceeding with MultinoimialNB.

- Key Metrics for success in solving problem under consideration

  To find out best performing model following metrices are used:

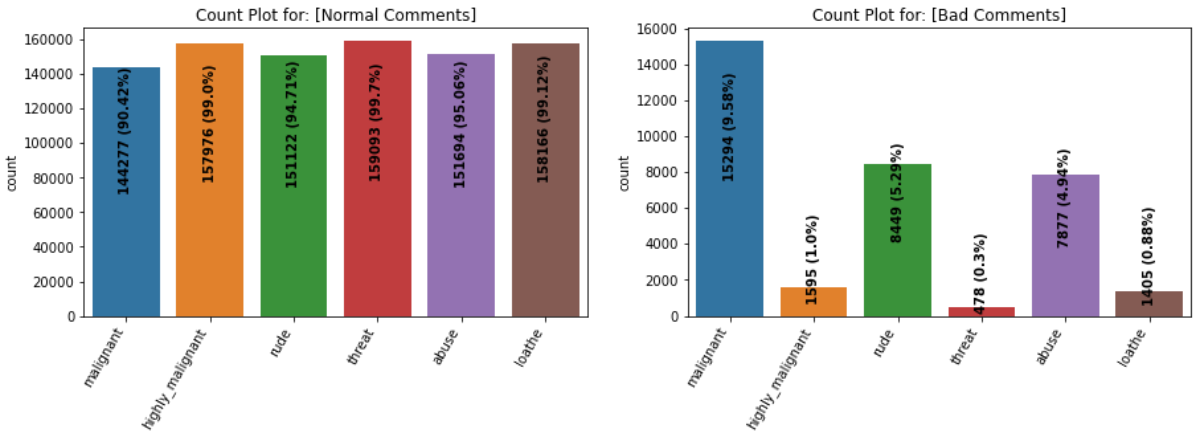
  1. Accuracy Score: It is used to check the model performance score between 0.0 to 1.0
  2. Hamming Loss: The Hamming loss is the fraction of labels that are incorrectly predicted.
  3. Classification Report: A Classification report is used to measure the quality of predictions from a classification algorithm. How many predictions are True and how many are False.

- Visualizations

  To better understand the data, following types of visualizations have been used: 1. Univariate.

  1. Univariate Analysis: Univariate analysis is the simplest form of data analysis where the data being analysed contains only one variable. In this project, distribution plot, count plot and box plot has been used.

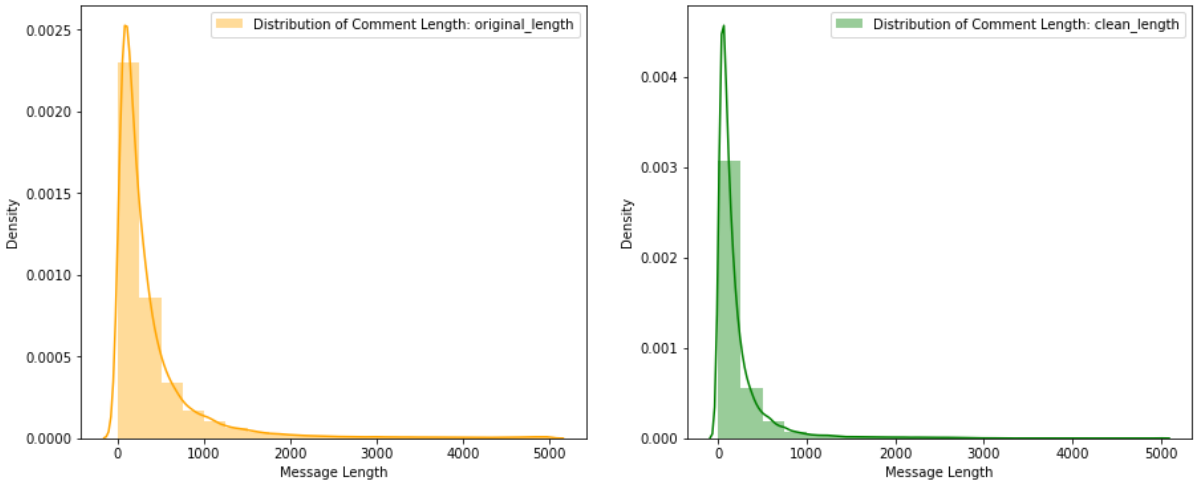
Count Plot (countplot):



Remarks:

- Dataset consists of higher number of **`Normal Comments`** than **`Bad or Malignant Comments`**. ***Therefore, it is clear that dataset is imbalanced and needs to be handle accordingly.**
- Most of the bad comments are of type `malignant` while least number of type `threat` is present in dataset.
- Majority of bad comments are of type `malignant`, `rude` and `abuse`.

Distribution Plot (distplot):



Remarks:

- Before cleaning comment text, most of the comment's length lies between 0 to 1100 while after cleaning, it lies between 0 to 900.

Displaying with WordCloud (wordcloud):

Remarks:

- From wordcloud of `malignant` comments, it is clear that it mostly consists of words like fuck, nigger, moron, hate, suck ect.
- From wordcloud of `highly_malignant` comments, it is clear that it mostly consists of words like ass, fuck, bitch, shit, die, suck, faggot ect.
- From wordcloud of `rude` comments, it is clear that it mostly consists of words like nigger, ass, fuck, suck, bullshit, bitch etc.
- From wordcloud of `threat` comments, it is clear that it mostly consists of words like die, must die, kill, murder etc.
- From wordcloud of `abuse` comments, it is clear that it mostly consists of words like moron, nigger, fat, jew, bitch etc.
- From wordcloud of `loathe` comments, it is clear that it mostly consists of words like nigga, stupid, nigger, die, gay cunt etc.

- # Interpretation of the Results

  Starting with univariate analysis, with the help of count plot it was found that dataset is imbalanced with having higher number of records for normal comments than bad comments (including malignant, highly malignant, rude, threat, abuse and loathe). Also, with the help of distribution plot for comments length, it was found

that after cleaning, most of comments length decreases from range 0-1100 to 0-900. Moving further with wordcloud, it was found that malignant comments consists of words like fuck, nigger, moron, hate, suck ect., highly_malignant comments consists of words like ass, fuck, bitch, shit, die, suck, faggot ect., rude comments consists of words like nigger, ass, fuck, suck, bullshit, bitch etc., threat comments consists of words like die, must die, kill, murder etc., abuse comments consists of words like moron, nigger, fat, jew, bitch etc. and loathe comments consists of words like nigga, stupid, nigger, die, gay cunt etc.

CONCLUSION

- Key Findings and Conclusions of the Study

  From the above model comparision it is clear that MultinomialNB performs better with **Accuracy Score: 90.74%** and **Hamming Loss: 2.4%** than other models. Therefore, proceeding with MultinoimialNB.

- Learning Outcomes of the Study in respect of Data Science

  During the data analysis, comment_text feature contains text as well as numbers but I have only taken text and discarded the numbers. But these numbers can also be replaced with some other text which might impact the model performance either in positive or negative way. As of now, I am finishing this project with my current approach which gives the **final accuracy score of 90.74% and hamming loss: 2.4%** and this can be further improved by training with more specific data.

- Limitations of this work and Scope for Future Work

Current model is limited to comments text data but this can further be improved for other sectors of foul language detection by training the model accordingly. The overall score can also be improved further by training the model with more specific data.