FLIP ROBO

Malignant Comments Classifier Project

NAME OF THE PROJECT

Salman Pradhan

Submitted by:

Salman Pradhan

ACKNOWLEDGMENT

I wish to thank Flip Robo Technologies for assigning this project to me and thanks to Mr.Shubham Yadav sir for his valuable technical support
on this project.

INTRODUCTION

- Business Problem Framing

We have to build an application in order to predict 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

  Created model can be used to predict online hate and abuse comment so that it can be controlled and restricted from spreading hatred and cyberbullying.

- Review of Literature

Now a days people use internet and more.They can easily share their views on products or persons . 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.

- Motivation for the Problem Undertaken

DataScience help us to make predictions at areas like health sectors, auto industry, education, media etc. For our project we decided to classify internet comments.

Analytical Problem Framing

- Data Sources and their formats

We have two dataset one is Train.csv and other is Test.csv.

Train.csv has 159171 rows and 8 columns. And Test.csv has 153164 rows.

```
df=pd.read_csv("C:/Users/Lenovo/OneDrive/Desktop/Malignant Comments Classifier Project/train.csv")
df
```

	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
...	...	...	...	...	...	...	...	...
159566	ffe987279560d7ff	":::::And for the second time of asking, when ...	0	0	0	0	0	0
159567	ffea4adeee384e90	You should be ashamed of yourself \n\nThat is ...	0	0	0	0	0	0
159568	ffee36eab5c267c9	Spitzer \n\nUmm, theres no actual article for ...	0	0	0	0	0	0
159569	fff125370e4aaaf3	And it looks like it was actually you who put ...	0	0	0	0	0	0
159570	fff46fc426af1f9a	"\nAnd ... I really don't think you understand...	0	0	0	0	0	0

159571 rows × 8 columns

```
#checking dimension
df.shape
```

```
(159571, 8)
```

```python
df1=pd.read_csv("C:/Users/Lenovo/OneDrive/Desktop/Malignant Comments Classifier Project/test.csv")
df1
```

	id	comment_text
0	00001cee341fdb12	Yo bitch Ja Rule is more succesful then you'll...
1	0000247867823ef7	== From RfC == \n\n The title is fine as it is...
2	00013b17ad220c46	" \n\n == Sources == \n\n * Zawe Ashton on Lap...
3	00017563c3f7919a	:If you have a look back at the source, the in...
4	00017695ad8997eb	I don't anonymously edit articles at all.
...	...	...
153159	fffcd0960ee309b5	. \n i totally agree, this stuff is nothing bu...
153160	fffd7a9a6eb32c16	== Throw from out field to home plate. == \n\n...
153161	fffda9e8d6fafa9e	" \n\n == Okinotorishima categories == \n\n I ...
153162	fffe8f1340a79fc2	" \n\n == ""One of the founding nations of the...
153163	ffffce3fb183ee80	" \n :::Stop already. Your bullshit is not wel...

153164 rows × 2 columns

- Data Preprocessing Done

1. First we have to remove all other words and symbols except alphates (a-z and A-Z).
2. We have to make all the words lower.
3. We have to split words from different sentences.
4. Then we have made a list named stoplist which contains all the repetation of words whose impact is neglisible while predicting the output.
5. We have to remove all the punctuations that present inside the sentence.
6. After removing all these unnecessary noise ,we have converted words to their base form by using WordnetLemmatizer.
7. Aproxmately 20000000 unnecessary words are removed.
8. We remove large sentence whose Length of words more than 400 .

```python
#Data preprocessing
wordnet=WordNetLemmatizer()
corpus = []#remove noise and punctuation
for i in range(0,153164):
    review = re.sub("[^a-zA-z\s']", ' ', df1['comment_text'][i])
    review = review.lower()
    review = review.split()
    review = [wordnet.lemmatize(word) for word in review if word not in stoplist if word not in punc]
    review = ' '.join(review)
    corpus.append(review)
```

```
df1=df1[df1['len']<400]#remove long text
df1
```

- State the set of assumptions (if any) related to the problem under consideration

  We remove large sentence whose Length of words more than 400.

- Hardware and Software Requirements and Tools Used

  1.NumPy: Base n-dimensional array package
  2.Matplotlib: Comprehensive 2D/3D plotting
  3Seaborn: For plotting graph
  4.Pandas: Data structures and analysis
  5.Nltk:for converting text to vectors
  6.String: For data cleaning
  7.Re: For data cleaning and remove noise from dataset
  8.Scikit-multilearn: provides a range of algorithm

Model/s Development and Evaluation

- Identification of possible problem-solving approaches (methods)

  These are the approaches to solve this problem
  1.Understand business problem
  2.Data analysis
  3.Data Cleaning
  4.Data preprocessing
  5.Model Building
  6. Model Evaluation
  7. Selecting the best model
  8.Predict test dataset

- # Testing of Identified Approaches (Algorithms)

  Algorithms used for this dataset are
  1. MLkNN
  2. BinaryRelevance
  3. ClassifierChain
  4. LabelPowerset

- # Run and Evaluate selected models

```
mlknn_classifier = MLkNN()
mlknn_classifier.fit(x_train, y_train)
```

```
MLkNN(ignore_first_neighbours=0, k=10, s=1.0)
```

```
y_pred = mlknn_classifier.predict(x_test)
print('Test accuracy is {}'.format(accuracy_score(y_test,y_pred)))
print('Test classification report is {}'.format(classification_report(y_test,y_pred)))
```

```
Test accuracy is 0.8845323430348044
Test classification report is               precision    recall  f1-score   support

           0       0.78      0.31      0.44      2592
           1       0.60      0.11      0.19       264
           2       0.73      0.37      0.49      1464
           3       0.50      0.10      0.16        72
           4       0.65      0.35      0.46      1396
           5       0.60      0.10      0.18       254

   micro avg       0.72      0.31      0.44      6042
   macro avg       0.64      0.22      0.32      6042
weighted avg       0.72      0.31      0.43      6042
 samples avg       0.03      0.03      0.03      6042
```

```
# using binary relevance
from skmultilearn.problem_transform import BinaryRelevance
from sklearn.naive_bayes import GaussianNB
# initialize binary relevance multi-label classifier
# with a gaussian naive bayes base classifier
classifier = BinaryRelevance(GaussianNB())
# train
classifier.fit(x_train, y_train)
# predict
y_pred = classifier.predict(x_test)
# accuracy
print("Accuracy = ",accuracy_score(y_test,y_pred))
print('Test classification report is {}'.format(classification_report(y_test,y_pred)))
```

```
Accuracy =  0.33141894809569045
Test classification report is               precision    recall  f1-score   support

           0       0.18      0.94      0.30      2592
           1       0.04      0.68      0.08       264
           2       0.11      0.91      0.19      1464
           3       0.01      0.44      0.03        72
           4       0.10      0.91      0.18      1396
           5       0.03      0.65      0.06       254

   micro avg       0.11      0.90      0.19      6042
   macro avg       0.08      0.75      0.14      6042
weighted avg       0.13      0.90      0.23      6042
 samples avg       0.06      0.10      0.07      6042
```

```
# initialize classifier chains multi-label classifier
classifier = ClassifierChain(LogisticRegression())
# Training logistic regression model on train data
classifier.fit(x_train, y_train)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/_logistic.py:940: ConvergenceWarning: lbfgs failed to converge (sta
tus=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
```

```
ClassifierChain(classifier=LogisticRegression(C=1.0, class_weight=None,
                                              dual=False, fit_intercept=True,
                                              intercept_scaling=1,
                                              l1_ratio=None, max_iter=100,
                                              multi_class='auto', n_jobs=None,
                                              penalty='l2', random_state=None,
                                              solver='lbfgs', tol=0.0001,
                                              verbose=0, warm_start=False),
                order=None, require_dense=[True, True])
```

```
# predict
y_pred= classifier.predict(x_test)
# accuracy
print("Accuracy = ",accuracy_score(y_test,y_pred))
print('Test classification report is {}'.format(classification_report(y_test,y_pred)))
```

```
Accuracy =  0.9100094999568183
```

```
classifier2 = LabelPowerset(LogisticRegression())
# train
classifier2.fit(x_train, y_train)
# predict
y_pred = classifier2.predict(x_test)
# accuracy
print("Accuracy = ",accuracy_score(y_test,y_pred))
print('Test classification report is {}'.format(classification_report(y_test,y_pred)))
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/_logistic.py:940: ConvergenceWarning: lbfgs failed to converge (sta
tus=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
```

```
Accuracy =  0.9069004231798946
Test classification report is               precision    recall  f1-score   support

           0       0.94      0.57      0.71      2592
           1       0.42      0.11      0.18       264
           2       0.90      0.64      0.75      1464
           3       0.55      0.08      0.14        72
           4       0.78      0.54      0.63      1396
           5       0.70      0.15      0.25       254

   micro avg       0.87      0.53      0.66      6042
   macro avg       0.71      0.35      0.44      6042
weighted avg       0.86      0.53      0.65      6042
 samples avg       0.06      0.05      0.05      6042
```

- Key Metrics for success in solving problem under consideration

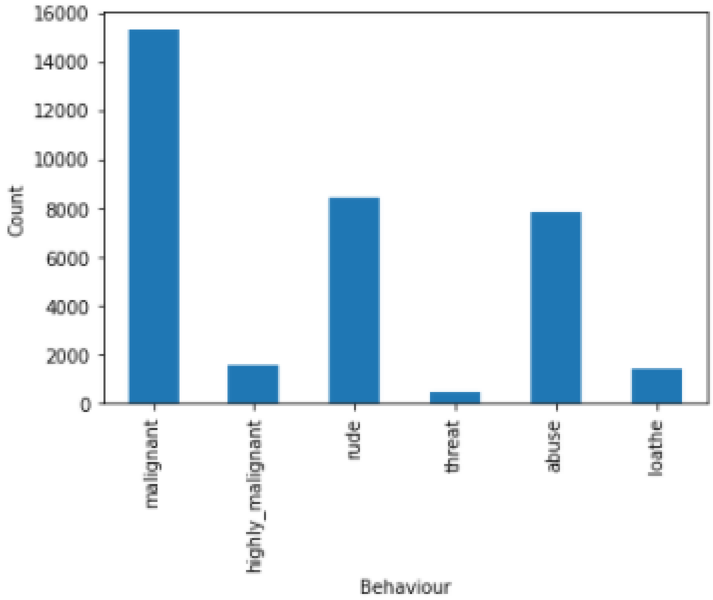
  Multi-label classifier is the best model for this dataset.

- Visualizations

  Count of comment classification

```
train_labels = df[['malignant', 'highly_malignant', 'rude', 'threat',
       'abuse', 'loathe']]
label_count = train_labels.sum()
```

```
label_count.plot(kind='bar')
plt.xlabel("Behaviour")
plt.ylabel("Count")
plt.show()
```

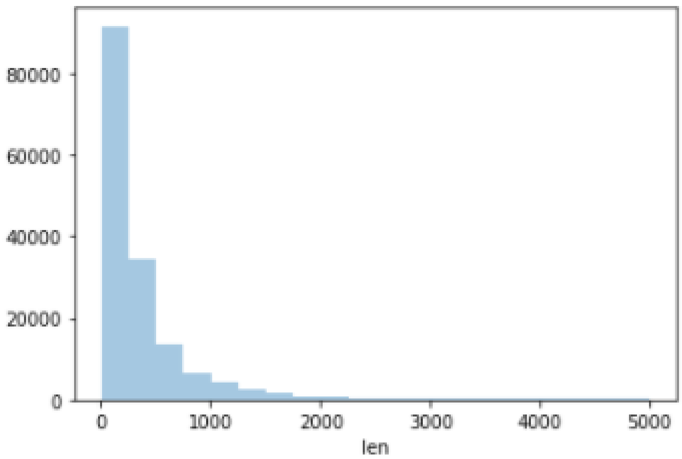


malignant comment are more and threat comments are less

```
]: sns.distplot(df['len'], kde=False, bins=20)
```

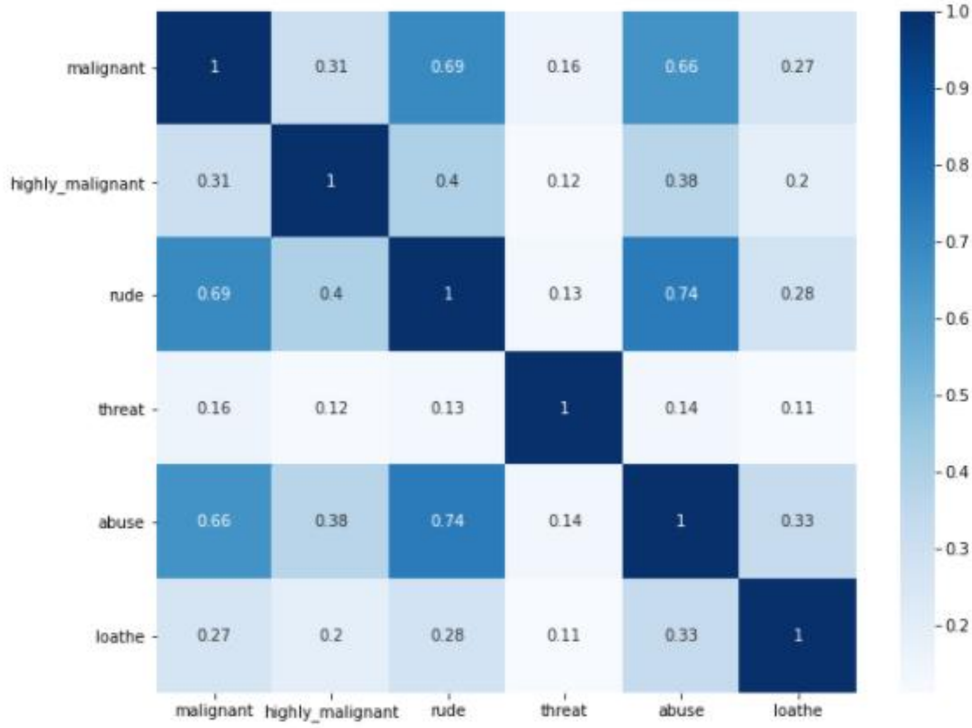
```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2557: FutureWarning
ll be removed in a future version. Please adapt your code to use either `displot` (
lity) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

```
]: <matplotlib.axes._subplots.AxesSubplot at 0x7f25325a7610>
```



Most Comment’s length is between 0 to 250 letters.

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f253252f2d0>
```



Abuse and rude are more corelated with each other.

- Interpretation of the Results

Comparing recall,precision,f1score of all models,we choose multi-label classifier is our best model.It gives accuracy score of approximately 91%.

Then we predict output of Test.csv dataset.

CONCLUSION

- Key Findings and Conclusions of the Study

1. First we load datasets Train.csv and Test .csv. 2. Then we preprocess data of both train.csv and test.csv by cleaning duplicates,noise and unnecessary words which are not helpful for this project

3. Then we convert text into vectors by using tfidfvectorizer as we know our ml model understands only integers.

4.We remove the rows whose length is more than 400.

5.Then we build our model.

5. We clearly see that f1 score of multi-label classifierr is good compared to all other algorithms.

6. We confirm and conclude that this is the best model.we predict test.csv

7. We save the model by using joblib for future use.

- Learning Outcomes of the Study in respect of Data Science

Multi-label classifier is the best model for the dataset.