CPSC-5310-A: Data Mining

Final Project : Naive Bayes Classification

Course Teahcer : Professor Dr. Wendy Osborn
Submitted by : Rajib Chandra Das
ID : 001201973

December 3, 2017

Contents

A Prediction to Teenager Suicide Ideation on the Perspective of Bangladesh

It was proposed that we will classify the special text of social media, more specifically words that are related with suicidal, mental illness. And we were planned to implement Naive Bayes classifier. So, now we are going to describe our implemented technique step by step.

1 Components that implemented

There are many components we have used, some important of them are given below.

Data Collection: our biggest challenge was collecting the special kind of text data from social media. We collected data two sub groups of reddit (subreddit): Suicide Watch (r/SuicideWatch) [1] and mentalhealth(r/mentalhealth) [2]. Munmun et. al. [3] also worked with reddit data, they could manage an API getting all posts and comments from these two sub reddits. At first we also tried to conduct API, but for technical issues we could not fin the API, hence we needed to copy relevant posts and comments manually. We created 200 hundreds file for train the system and another 100 for testing purpose.

Data Cleaning and preparing in this stage we only have the raw data collected from reddit which need to be cleaned. For cleaning purpose, we wrote a python code (stopWordsRemover.py) which can remove all stop words (a, and, but, how, what, when, who, . , etc). Here we also remove all words which length is less than four because there are very rare words that can be both at the same time, length below four and mental illness or suicidal word.

Then we made a dictionary to store the word that are most commonly occurred.

Feature extraction : for extracting features of data. we assign a label for every word in dictionary and then produce a frequency matrix, number of occurring the word in the train or test corpus.

Naive Bayes model computation: Here we have implemented three types of Naive Bayes classifier. Gaussian, Multinomial and Bernoulli Naive Bayes model to compute the accuracy. For different scenario, different Naive Bayes classifier model is performed better.

2 Evaluation Strategy

We evaluated our implemented Naive Bayes model(s) by comparing the score with existing method. Me and Khairul Bashar also conducted same project with same data but by different approach. we also checked our system with the testing data of each other. Besides this, we would compare accuracy of our system with Kim et. al. [4] because in their works they have proposed an efficient Naive Bayes classifier. Four evaluation criteria are given below:

Gaussian, multinomial, Bernoulli : we invoked this classifier from sklearn naive Bayes.

Decision trees : the score that Jahid achieved.
And for all of the models we determine following scores:

precision : positive predictive value.

recall : Sensitivity or true positive value.

f1-score : weighted average of the precision and recall. We evaluated f1-macro and f1-micro (details in Results) with [4].

3 Results

Here we have enclosed some screenshots that we have found for different conditions. If you want to run our application then follow the following procedure:

- (If necessary) Install scikit-learn, numpy, scipy. for python 2.7 and nltk for 3
- From terminal go to folder that contains our codes (DM_Project_Rajib)
- Run naiveBayesMain.py by the following command: **python2.7.13 naiveBayesMain.py**
- Output will be displayed for three different naive Bayes model.

So, now we are going to illustrate step by step to show the results. At first, if you want to remove stop words from all files of a folder, then copy **stopWordsRemover.py & tempOutputFile.txt** to the folder. From terminal go to that folder and run stopWordsRemover.py by following command: **python3.6 stopWordsRemover.py**. Then delete **stopWordsRemover.py & tempOutFile.txt** files from that folder.

Here, we have remove the stop words in cleaning stage and stored the clean train data in **train_corpus_clean** and test data in **test_corpus_clean**. And in

train_corpus and **test_corpus** raw data are stored for train and test corpus respectively.

So, now there are four cases,

Case 1: Default User can check the result for raw train and test data. In that case, he/she should run the following command: **python2.7.13 naiveBayesMain.py**. For details look at the screenshot below:

```
~/Desktop/DM_Project_Rajib$ python2.7.13 naiveBayesMain.py

**************************************
Gaussian Naive Bayes Model:
Accuracy Score:  0.960396039604
Precision-score :  0.925925925926
Recall-score :  1.0
f1-score (macro) :  0.960361067504
f1-score (micro):  0.960396039604

Multinomial Naive Bayes Model:
Accuracy Score:  0.613861386139
Precision-score :  0.823529411765
Recall-score :  0.28
f1-score (macro) :  0.564510779436
f1-score (micro):  0.613861386139

Bernoulli Naive Bayes Model:
Accuracy Score:  0.940594059406
Precision-score :  0.892857142857
Recall-score :  1.0
f1-score (macro) :  0.940448113208
f1-score (micro):  0.940594059406

**************************************
~/Desktop/DM_Project_Rajib$
```

Figure 1: output for both of train and test Unclean data.

From the above screenshot, we can see that for unclean data we found 96.03%, 61.38% and 94.05% using Gaussian, Multinomial and Bernoulli respectively.

Case 2: You can check the result for clean train but raw (unclean) test data. In that case, you should run the following command: **python2.7.13 naiveBayesMain.py train_corpus_clean**. Details are shown below:

```
~/Desktop/DM_Project_Rajib$ python2.7.13 naiveBayesMain.py  train_corpus_clean

************************************
Gaussian Naive Bayes Model:
Accuracy Score:  0.831683168317
Precision-score :  0.923076923077
Recall-score :  0.72
f1-score (macro) :  0.829273143084
f1-score (micro):  0.831683168317

Multinomial Naive Bayes Model:
Accuracy Score:  0.613861386139
Precision-score :  0.823529411765
Recall-score :  0.28
f1-score (macro) :  0.564510779436
f1-score (micro):  0.613861386139

Bernoulli Naive Bayes Model:
Accuracy Score:  0.920792079208
Precision-score :  0.875
Recall-score :  0.98
f1-score (macro) :  0.920597484277
f1-score (micro):  0.920792079208

************************************
~/Desktop/DM_Project_Rajib$
```

Figure 2: output for train data clean but test is not.

From the above screenshot, for case 2, we found 83.16%, 61.38% and 92.07% using Gaussian, Multinomial and Bernoulli respectively.

Case 3: To check output for unclean train but clean test data. In that case, you should run the following command: **python2.7.13 naiveBayesMain.py test_corpus_clean**. For details look at the screenshot below:

```
~/Desktop/DM_Project_Rajib$ python2.7.13 naiveBayesMain.py  test_corpus_clean

************************************
Gaussian Naive Bayes Model:
Accuracy Score:  0.950495049505
Precision-score :  0.924528301887
Recall-score :  0.98
f1-score (macro) :  0.950475630087
f1-score (micro):  0.950495049505

Multinomial Naive Bayes Model:
Accuracy Score:  0.673267326733
Precision-score :  0.869565217391
Recall-score :  0.4
f1-score (macro) :  0.646065625996
f1-score (micro):  0.673267326733

Bernoulli Naive Bayes Model:
Accuracy Score:  0.940594059406
Precision-score :  0.892857142857
Recall-score :  1.0
f1-score (macro) :  0.940448113208
f1-score (micro):  0.940594059406

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

Figure 3: output for test data clean but train is not.

From the above screenshot, for case 3, we found 95.04%, 67.32% and 94.05% using Gaussian, Multinomial and Bernoulli respectively.

Case 4: To check the output for both clean train and test data, you should run the following command: **python2.7.13 naiveBayesMain.py train_corpus_clean (space) test_corpus_clean**. Details are shown below:

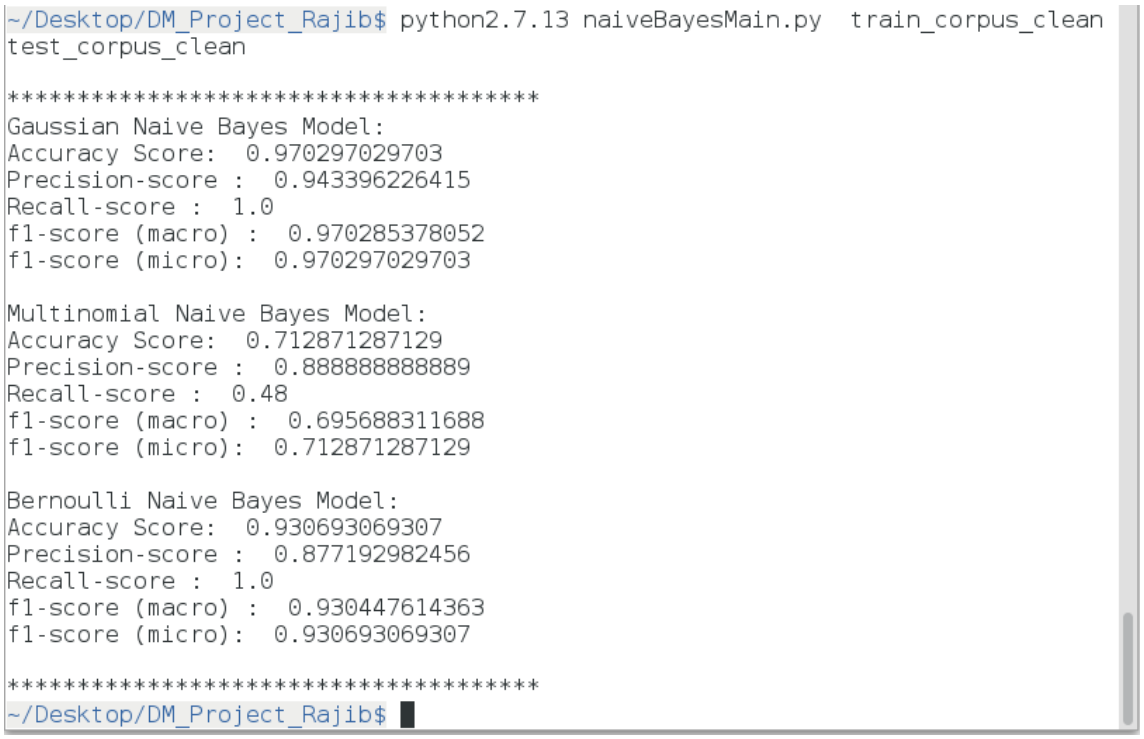

Figure 4: output for both of train and test clean data

From the above screenshot, for case 4, we found 97.03%, 71.28% and 93.07% using Gaussian, Multinomial and Bernoulli respectively.

On the contrary, with same training and test data Jahid achieved 95% accuracy by his decision tree classifier.

4 Conclusion

We found almost same result for Gaussian and Bernoulli model but for multinomial we got a gap in score when we tried for unclean data. But when we started to clean one by one we found improvement in accuracy for multinomial model. And we used clean data for both train and test data we found a handsome output in all models.

So, it can be concluded that, we need to clean the data very carefully. While we were cleaning the data, besides removing stop words we also used another technique

for not to count the words which length is less than four. That is why when we used clean data wee found our best result.

References

[1] Suicide Watch. https://www.reddit.com/r/SuicideWatch/.

[2] Mental Health. https://www.reddit.com/r/mentalhealth/.

[3] Munmun De Choudhury, Emre Kiciman, Mark Dredze, Glen Coppersmith, and Mrinal Kumar. Discovering shifts to suicidal ideation from mental health content in social media. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, pages 2098–2110. ACM, 2016.

[4] Sang-Bum Kim, Kyoung-Soo Han, Hae-Chang Rim, and Sung Hyon Myaeng. Some effective techniques for naive bayes text classification. *IEEE transactions on knowledge and data engineering*, 18(11):1457–1466, 2006.