Megha Vijendra
1001736938
mxv6938

Machine Learning
Project 2

Language used: Python
Name of the file: project1_mxv6983.py
Libraries used: os, math, random, timeit
To run file: python3 project2_mxv6983.py

Overview

A Naive Bayes classifier implemented from scratch using Bag of Words model and nive bayers probability formula to test the classification model.

Bayesian learning for classifying news text articles:
Naive Bayes classifiers are among the most successful known algorithms for learning to classify text documents.

Exploring the Dataset

The dataset 20newsgroup is made up of 20 sub folders of different fields of news articles. Each newsgroup or subfolder is made up of approximately 1000 files.

Data: http://www.cs.cmu.edu/afs/cs/project/theo-11/www/naivebayes.html (Newsgroup Data)

Following are the list of target folders which are used for classification:

['alt.atheism',
'comp.graphics',
'comp.os.ms-windows.misc',
'comp.sys.ibm.pc.hardware',
'comp.sys.mac.hardware',
'comp.windows.x',
'misc.forsale',
'rec.autos',
'rec.motorcycles',
'rec.sport.baseball',
'rec.sport.hockey',
'sci.crypt',
'sci.electronics',
'sci.med',
'sci.space',
'soc.religion.christian',
'talk.politics.guns',
'talk.politics.mideast',
'talk.politics.misc',
'talk.religion.misc']

To calculate the probabilities of each word, we iterate through each word in each file. Check, if that word was encountered before in the present newsgroup or class. If yes, then we increase the frequency count and if no, then we start counting.
Then we use the following formula for classification:

$$\underbrace{P(c\,|\,x)}_{\text{Posterior Probability}}=\frac{\overbrace{P(x\,|\,c)}^{\text{Likelihood}}\,\overbrace{P(c)}^{\text{Class Prior Probability}}}{\underbrace{P(x)}_{\text{Predictor Prior Probability}}}$$

Implementation of the Naïve Bayes Classifier

Here the problem statement is to classify a file to a particular target group.
The steps and the methods used in the implementation are as follows:

STEP 1: Splitting the data into Train and test set:

```
Method used: def train_test_split()
```

- Here the dataset of 1000 files of each category are split into training and test set.
- Here the requirement is to have equal split. Hence we have 500 files of each category for train and test sets.

STEP 2: Data pre-processing and cleaning:

```
Method used: def clean_data(contents):
```

- In the pre-processing stage, the data is cleaned by the clean_data user defined function in the program.
- All the uppercase letters are converted to lowercase ones in the contents.
- There are some characters like \n and \t that are necessary to either be removed or replaced with space character (so that the words can later be split as a list of words).
- All the characters inside the chars from the data, are replaced with space. The following list chars list is considered:
  chars = "\\`*_{}[]()>#+-.!$^'!/=,:\"<?|-#*_@"
- After replacing the special characters and converting to lower, here we are going to remove the stop words from the data. The commonly occurring words (stop-words) that do not contribute to the classification of the data needs to be removed.

STEP 3: Train the model using bag of words:

```
Method used: def bag_of_words(train_groups):
```

- In the training stage, the random 500 files of each class/newsgroup are selected and the file data is split with character space by pre-processing.
- Here, the number of occurrences of each word in each class file is counted
- A dictionary is created to maintain the count of each word.

- After the training of the data, there are around 170,000 different words in all classes (for 10,000 files).
- The bag of words model is used to keep count of all the words from each target folder, which will be later used to find the probability.

STEP 3: Test the model using Naïve Bayes:

```
Method used: def naive_bayers(test_groups,words_dict,total_count_dict),
             def calc_props(words_dict, class_name, contents, len_classes)
```

- In the testing stage, after the training stage the files are selected randomly from the remaining files and are opened one after the other.
- And the similar approach is implemented with the testing data. The files are cleaned with clean_data function, and post pre-processing the probability of each word in these files are determined for all the classes/newsgroups
- The following is used to calculate the posterior probability of a particular file. We are doing Laplacian smoothening for new words with 0 count frequency. The smoothening peaks at 0.001.

```
For each newsgroup n:
{ Pn = 0
    For each word in preprocessed_file:
    {
            Pn = Pn + log(P(word| n))
    }
}
```

- The function calc_ props compute the probability for a given class/newsgroup. The predicted class would be the one has the highest probability Pn.

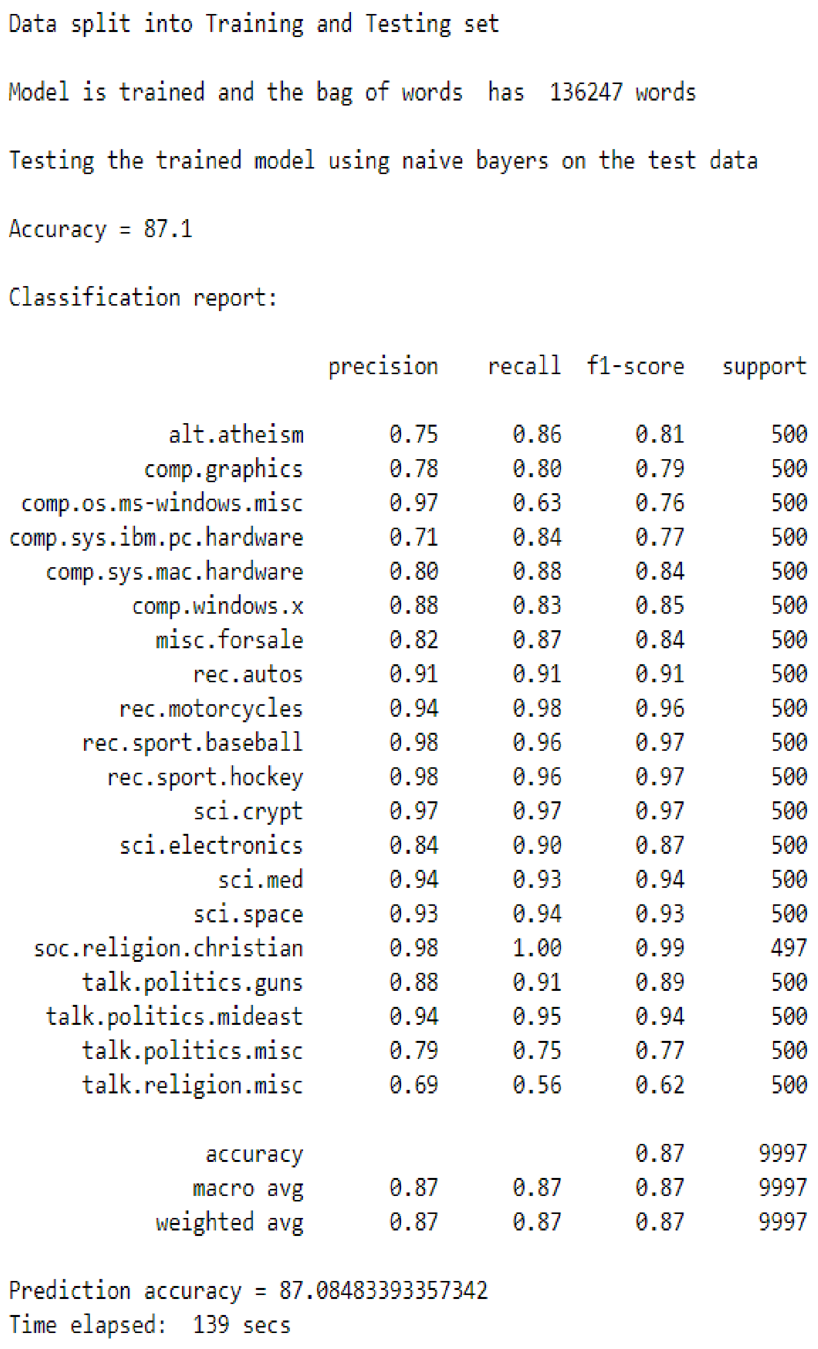
Results and Analysis

- As you can see from the results summary table, the Naïve Bayes Classifier model best performs with using at least half the dataset (500 files each class) as training data in terms of accuracy.
- Therefore, the reason behind giving the split value is as 500 in the problem statement because in case of small datasets, each dataset must contain some good amount of data,
- Having less data in the dataset means that the data will not be trained well enough and the that does not serve the model accuracy well.
- The following results table shows the accuracy of the model against different training dataset sizes. And we can see that after 500 the accuracy is almost the same.

TRAINING SET	ACCURACY	TIME
900	87.7755511022044	128 secs
750	87.9103282626101	139 secs
500	87.08483393357342	309 secs

400	86.67889296432143	280 secs
250	84.15577487329955	190 secs
150	82.06048481995764	177 secs
50	75.98968203832386	176 secs

- The following diagram shows the classification report for the model when the training set = 500

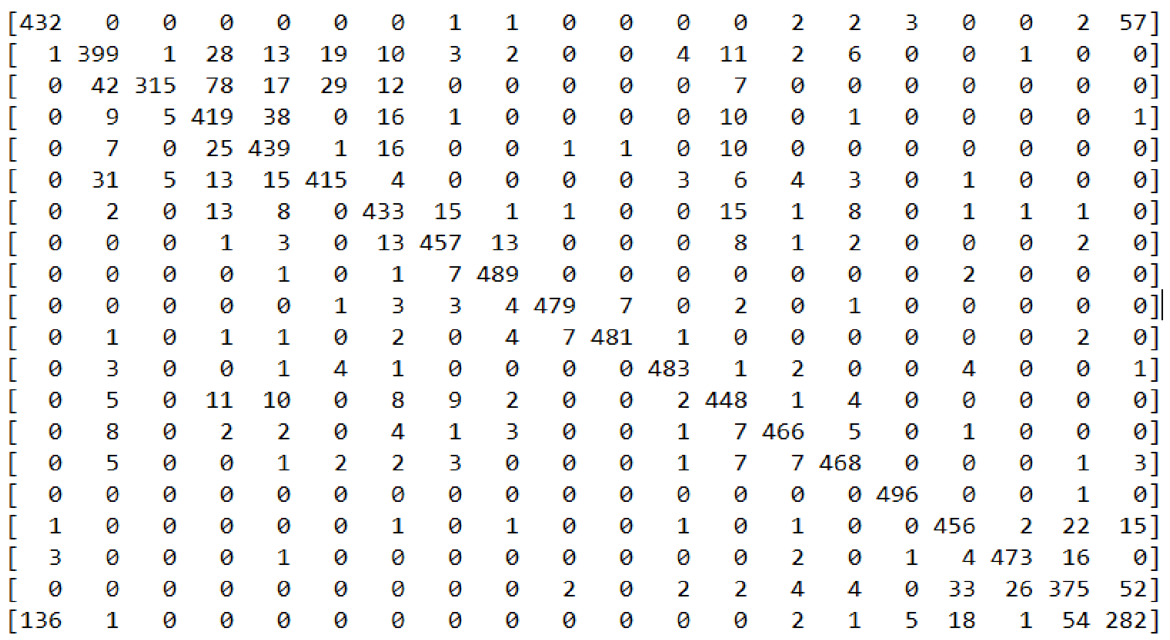


```
Data split into Training and Testing set

Model is trained and the bag of words  has  136247 words

Testing the trained model using naive bayers on the test data

Accuracy = 87.1

Classification report:

                          precision    recall  f1-score   support

             alt.atheism       0.75      0.86      0.81       500
           comp.graphics       0.78      0.80      0.79       500
 comp.os.ms-windows.misc       0.97      0.63      0.76       500
comp.sys.ibm.pc.hardware       0.71      0.84      0.77       500
   comp.sys.mac.hardware       0.80      0.88      0.84       500
          comp.windows.x       0.88      0.83      0.85       500
            misc.forsale       0.82      0.87      0.84       500
               rec.autos       0.91      0.91      0.91       500
         rec.motorcycles       0.94      0.98      0.96       500
      rec.sport.baseball       0.98      0.96      0.97       500
        rec.sport.hockey       0.98      0.96      0.97       500
               sci.crypt       0.97      0.97      0.97       500
         sci.electronics       0.84      0.90      0.87       500
                 sci.med       0.94      0.93      0.94       500
               sci.space       0.93      0.94      0.93       500
  soc.religion.christian       0.98      1.00      0.99       497
      talk.politics.guns       0.88      0.91      0.89       500
   talk.politics.mideast       0.94      0.95      0.94       500
      talk.politics.misc       0.79      0.75      0.77       500
      talk.religion.misc       0.69      0.56      0.62       500

                accuracy                           0.87      9997
               macro avg       0.87      0.87      0.87      9997
            weighted avg       0.87      0.87      0.87      9997

Prediction accuracy = 87.08483393357342
Time elapsed:  139 secs
```

- Next, we are going to analyse the results using confusion matrix.
- A confusion matrix is a table that is often used to describe the performance of a classification model (or "classifier") on a set of test data for which the true values are known.

The predicted classes are represented in the columns of the matrix, whereas the actual classes are in the rows of the matrix. We then have four cases for each newsgroup in the data:
1. True positives (TP): the cases for which the classifier predicted 'correct class' and the actual data was from same class.
2. True negatives (TN): the cases for which the classifier did not predicted class as 'correct' and the actual data was also not from same class.
3. False positives (FP): the cases for which the classifier predicted class as 'correct class' but the actual data had a different correct class.
4. False negatives (FN): the cases for which the classifier did not predicted class as 'correct class' and the actual data was rather a correct class.

- The following diagram represents the confusion matrix when training set is 500.



```
[432   0   0   0   0   0   0   1   1   0   0   0   0   2   2   3   0   0   2  57]
[  1 399   1  28  13  19  10   3   2   0   0   4  11   2   6   0   0   1   0   0]
[  0  42 315  78  17  29  12   0   0   0   0   0   7   0   0   0   0   0   0   0]
[  0   9   5 419  38   0  16   1   0   0   0   0  10   0   1   0   0   0   0   1]
[  0   7   0  25 439   1  16   0   0   1   1   0  10   0   0   0   0   0   0   0]
[  0  31   5  13  15 415   4   0   0   0   0   3   6   4   3   0   1   0   0   0]
[  0   2   0  13   8   0 433  15   1   1   0   0  15   1   8   0   1   1   1   0]
[  0   0   0   1   3   0  13 457  13   0   0   0   8   1   2   0   0   0   2   0]
[  0   0   0   0   1   0   1   7 489   0   0   0   0   0   0   0   2   0   0   0]
[  0   0   0   0   0   1   3   3   4 479   7   0   2   0   1   0   0   0   0   0]
[  0   1   0   1   1   0   2   0   4   7 481   1   0   0   0   0   0   0   2   0]
[  0   3   0   0   1   4   1   0   0   0   0 483   1   2   0   0   4   0   0   1]
[  0   5   0  11  10   0   8   9   2   0   0   2 448   1   4   0   0   0   0   0]
[  0   8   0   2   2   0   4   1   3   0   0   1   7 466   5   0   1   0   0   0]
[  0   5   0   0   1   2   2   3   0   0   0   1   7   7 468   0   0   0   1   3]
[  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 496   0   0   1   0]
[  1   0   0   0   0   0   1   0   1   0   0   1   0   1   0   0 456   2  22  15]
[  3   0   0   0   1   0   0   0   0   0   0   0   0   2   0   1   4 473  16   0]
[  0   0   0   0   0   0   0   0   0   2   0   2   2   4   4   0  33  26 375  52]
[136   1   0   0   0   0   0   0   0   0   0   0   0   2   1   5  18   1  54 282]
```

REFERENCES:

[1] https://www.analyticsvidhya.com/blog/2014/11/text-data-cleaning-steps-python/

[2] https://machinelearningmastery.com/naive-bayes-classifier-scratch-python/#:~:text=Naive%20Bayes%20is%20a%20classification,to%20make%20their%20calculations%20tractable.

[3] https://krakensystems.co/blog/2018/text-classification

[4] https://machinelearningmastery.com/classification-as-conditional-probability-and-the-naive-bayes-algorithm/

[5] https://stackoverflow.com/questions/43390697/python-3-creating-a-dictionary-adding-values-to-a-nested-dictionary