PROJECT 2 REPORT MACHINE LEARNING -6363

Name: Sanket Manik Salunke

Student Id: 1001764897

This project is based on implementation of Naïve Bayes Classifier

CODE Language Used – Python

Environment- Anaconda(Jupyter Notebook):

Note: If nltk stopwords are not installed in your system then please run below commands as I am using 'nltk' package to generate stopwords just to compare results.

Please refer the link if you get any issue: https://stackoverflow.com/questions/41610543/corpora-stopwords-not-found-when-import-nltk-library

```
>>import nltk

>>nltk.download('stopwords')
```

Steps performed while implementation are as follows:

- **Data Analysis:**
  As per my observation data belongs to textbook like structure but data has random digits, symbols etc. which needs to be remove before training the model.
- **Pre-processing:**
  - Given data has 20 different folders and inside those folders there were files so total files are around 1997. I am using following packages to peprocess the given data:
    Packages and usage explanation:

    ```
    import os  # to access the current working directory
    import math #using for sum calculation
    import random #using to generate random number in preprocessing
    from sklearn.metrics import confusion_matrix  # Just to print the confusion matrix
    import copy # copy the data
    import re # to implement regex in preprocessing- eliminate unwanted data
    ```

  - I am taking the data path where **20_newsgroups** folder is present to access the data for further data processing & implementation.
  - In pre-processing I am considering stopwords with all the frequent words and symbols which has been used for cleaning the data.
  - As per analysis, accuracy result depends on the data cleaning process and I will be explaining that in my conclusion.
    Data Pre-processing:

```
#https://stackoverflow.com/questions/12851791/removing-numbers-from-string
def preprocess(data):
    #removing digits
    data = ''.join([i for i in data if not i.isdigit()])


    #https://stackoverflow.com/questions/12628958/remove-small-words-using-python
    shortword = re.compile(r'\W*\b\w{1,3}\b')
    #remove symbols which are repeating
    test = re.compile(r'(.)\1{9,}')
    data = shortword.sub('', data)
    data = test.sub('', data)
    words = data.split(' ')
    return words
```

- After pre- processing and removing all the stopwords and unnecessary data I am calculating bag of words to get the total number of words and their count. Bag of words gives clear understanding of the words along with their occurrence in the file.
- As per the Naïve Bayes – which is based on the Bayes theorem we need to calculate the conditional probability for occurrences of words along with the total number of words.
  Bayes Theorem:

$$P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)}$$

  Where P(A|B) is a conditional probability: the likelihood of event A occurring given that B is true.

- **Training the model**:
  After this I am training the model by giving 50% training data to create bag of words for the total data by reading all the files inside the **20_newsgroups** folder which gives me total data. After reading this whole data I am implementing bag of words logic to get the words and frequencies.

```
#dictionary of all the words
bag_dict
```

```
{'alt.atheism': {'xref': 131,
  'moder': 11,
  'news': 840,
  'pple': 336,
  'ndrew': 139,
  'gnus': 10,
  'usenet': 123,
  'cwru': 186,
  'spool': 21,
  'uunet': 167,
  'bmpcug': 20,
  'newsgro': 519,
  'subject': 590,
  'summ': 18,
  'books': 70,
  'ddresses': 10,
  'nyth': 128,
  'keyw': 15,
  'follow': 142,
```

  This gives me all the words as directory and which can be access for the particular folder as below:

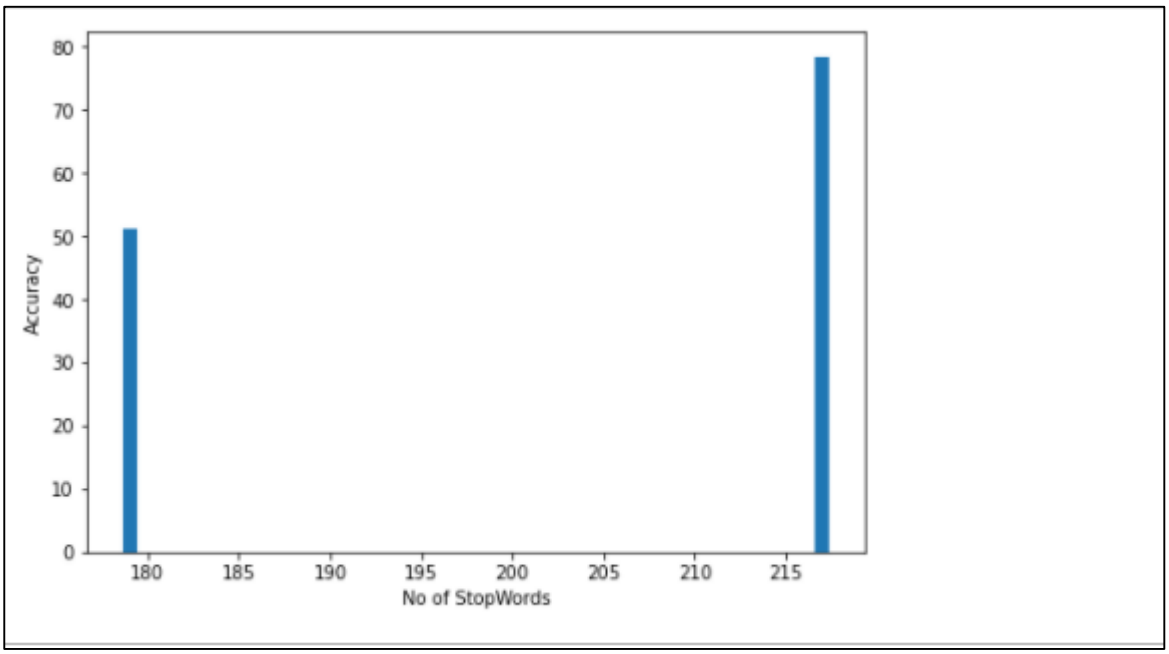
```
In [39]: bag_dict['comp.graphics']
Out[39]: {'xref': 171,
          'comp': 1482,
          'news': 761,
          'phod': 191,
          'newsgro': 556,
          'subject': 544,
          'present': 34,
          'reply': 102,
          'follow': 98,
          'rderock': 6,
          'nswc': 6,
          'tuesd': 4,
          'june': 6,
          'surf': 75,
          'center': 94,
          'rese': 126,
          'ness': 31,
          'neer': 37,
          'system': 238,
```

- **Probability Calculation using Bayes Theorem**:
  Then I am calculating the probability of word by iterating through all the words to calculate the total number of words in the dictionary.
  And once word matches then I am calculating the occurrence of that word and adding it to the list.
  This probability gives the testing value with all the classes and then I am calculating maximum in all of the values to get the actual probability.

  So from this success of words I am calculating the accuracy of the model.

  As per my observation, while pre-processing using stopwords we can see that if we increase the stop words and clean the data more then accuracy increases .

  Please refer the graph for the same:



  First, I have trained the model using total 217 stopwords which gave more accuracy of 78.5%
  Then I have used 179 stopwords to train the model which gave me 51.30%
  So it's important to clean the data as per the need by data analysis and usage.
  We can increase the accuracy by analysing the data and cleaning the data.

I got the confusion matrix & Classification Report for both of them as follows:

```
Confusion Martix :

[[9429   10    6   32   47    6  131   53   95   49    3    9    0   20
     0   16    7    6    4   77]
 [ 593 5510  404 1037  785  350  690  228   81   36   37   34  144   21
    36    0    8    4    2    0]
 [ 587  952 5062 1466  866  173  588  131   56   38    0    1   48    8
    24    0    0    0    0    0]
 [ 570  815  408 6100 1143   85  515  120   44   44   30    9  110    0
     0    0    0    3    4    0]
 [ 628  868  373 1012 6131   16  676  111   36   26   10    9   88    7
     6    0    0    0    2    1]
 [ 538 1475  291  939  754 5327  311   91   76   33    0   29  104   10
     6    0   15    0    0    1]
 [ 727  948  334  944  765    6 5675  235   76   77   52    3  105   21
    14    0   16    0    2    0]
 [1141  655  314  708  709   61  561 5297  310   75   21    0   96   15
    24    0   12    0    0    1]
 [1232  729  111  696  644   25  456  466 5585    0   20    0   16   14
     6    0    0    0    0    0]
 [1339  662  359  669  532   28  683  406  256 4945   92    0   10    7
     6    0    4    0    2    0]
 [1302  650  352  654  397   12 1026  284  140  364 4804    9    0    0
     0    0    2    0    4    0]
 [1573  749  406  762  868  387  379  299  224   27    6 4267   18    7
    12    0   16    0    0    0]
 [ 774 1183  301 1254 1214   83 1188  626  149   49   38    6 3090    0
    40    0    0    3    2    0]
 [2621  649  258  344  828   16  653  675  548  228   72   42  125 2889
    18   10   12    3    6    3]
 [1476 1656  162  542  660   87  921  809  453  179   10   31  223  112
  2650    5   20    0    3    1]
 [3146  740  309  249   91    3  656  490  910  561  108    8   60  128
    11 2487    0    0    2    1]
 [3783  125  105  264  192    6  234 1576 1061   79  136  327   51   63
   103    2 1838    2   46    7]
 [5560  242   57  187  273   12  576  656  303  131   57  101   95   65
    96   74  189 1255   67    4]
 [4053  295  140  357  413   21  407 1312  499  335  105  236   59  175
   134   62  547   84  703   63]
 [6912  166  117  246  115   23  194  588  325  207   42   44   26   49
    42  168  302    9  134  291]]
```

```
.......Classification Report.....
                          precision    recall  f1-score   support

               alt.atheism       0.20      0.94      0.33     10000
             comp.graphics       0.29      0.55      0.38     10000
   comp.os.ms-windows.misc       0.51      0.51      0.51     10000
  comp.sys.ibm.pc.hardware       0.33      0.61      0.43     10000
     comp.sys.mac.hardware       0.35      0.61      0.45     10000
            comp.windows.x       0.79      0.53      0.64     10000
              misc.forsale       0.34      0.57      0.43     10000
                 rec.autos       0.37      0.53      0.43     10000
           rec.motorcycles       0.50      0.56      0.53     10000
        rec.sport.baseball       0.66      0.49      0.57     10000
          rec.sport.hockey       0.85      0.48      0.61     10000
                 sci.crypt       0.83      0.43      0.56     10000
           sci.electronics       0.69      0.31      0.43     10000
                   sci.med       0.80      0.29      0.42     10000
                 sci.space       0.82      0.27      0.40     10000
    soc.religion.christian       0.88      0.25      0.39      9960
        talk.politics.guns       0.62      0.18      0.28     10000
     talk.politics.mideast       0.92      0.13      0.22     10000
        talk.politics.misc       0.72      0.07      0.13     10000
        talk.religion.misc       0.65      0.03      0.06     10000

                  accuracy                           0.42    199960
                 macro avg       0.61      0.42      0.41    199960
              weighted avg       0.61      0.42      0.41    199960
```

```
Accuracy = 51.3

Confusion Martix :

[[5974  172  211 1619  432   21  601  257  139  246   35    0   54   10
    13   79   39    5   13   80]
 [ 730 3181  416 2437  557  279 1208  500  276  126   56    6   66   28
    62   15   35   14    4    4]
 [ 714  762 2898 2525  581  117 1327  441  221  156   70    0  120   14
    24    5   12    6    4    3]
 [ 685  783  476 5206  586  104 1158  395  171   98   64   18  170   28
    37   10    8    0    2    1]
 [ 775  684  439 2164 3936  100 1032  424  204   88   40    0   72   22
     9    0    4    1    5    1]
 [ 745 1323  441 2101  496 3227  685  349  193  154   50   45  132    7
    28    0   14    3    6    1]
 [ 662 1016  401 2035  552  121 4384  334  137   91   72    3  120   14
    16    5   24    6    5    2]
 [ 859  806  445 1748  567   12 1155 3788  272  138   23   13  120   15
    18    5    8    0    6    2]
 [ 842  865  455 1590  677   43  858  555 3895  126   20   18   28    7
     6    0   12    0    2    1]
 [ 923  661  490 1768  559   31 1459  461  306 3185   82    9   45    0
    12    0    4    3    2    0]
 [1042  646  491 1695  408   58 1285  797  149  546 2777    0   52   10
    12    0   20    1    8    3]
 [1014  346  843 2379  366  221  798  345  238   61   25 3172   51    0
     6   11  100    1   21    2]
 [ 608 1525  177  641 1283  128 1509  944   13   20   14   13 3093    8
    12    0    4    0    8    0]
 [1135  938  350 1822 1007   59 1339  597  393  371   70    9  151 1673
    20   20   32    6    4    4]
 [1137 1167  494 1906  552  291 1042  945  219  228   40    9  217   55
  1632   11   44    3    4    4]
 [2312  421  292 2270  289   42  735  629  438  438   57    3   67   60
    15 1844   28    6    5    9]
 [1534  561  462 1853  667   47  799 1156  745  433  160   85  150   75
    39   17 1125   13   64   15]
 [1993  504  346 1895  469  228 1624  819  296  379  134   31  143   25
    82   62  123  781   55   11]
 [1521  644  384 1915  931   64  918 1224  696  515   85   76   98   72
    68   50  277   72  341   49]
 [4024  382  332 1401  809   34  603  778  426  375   61    4   67   50
    34  132  188    6   97  197]]
```

```
.......Classification Report.....
                          precision    recall  f1-score   support

               alt.atheism       0.20      0.60      0.30     10000
             comp.graphics       0.18      0.32      0.23     10000
   comp.os.ms-windows.misc       0.27      0.29      0.28     10000
  comp.sys.ibm.pc.hardware       0.13      0.52      0.20     10000
     comp.sys.mac.hardware       0.25      0.39      0.31     10000
            comp.windows.x       0.62      0.32      0.42     10000
              misc.forsale       0.18      0.44      0.25     10000
                 rec.autos       0.24      0.38      0.29     10000
           rec.motorcycles       0.41      0.39      0.40     10000
        rec.sport.baseball       0.41      0.32      0.36     10000
          rec.sport.hockey       0.71      0.28      0.40     10000
                 sci.crypt       0.90      0.32      0.47     10000
           sci.electronics       0.62      0.31      0.41     10000
                   sci.med       0.77      0.17      0.27     10000
                 sci.space       0.76      0.16      0.27     10000
    soc.religion.christian       0.81      0.19      0.30      9960
       talk.politics.guns       0.54      0.11      0.19     10000
    talk.politics.mideast       0.84      0.08      0.14     10000
       talk.politics.misc       0.52      0.03      0.06     10000
       talk.religion.misc       0.51      0.02      0.04     10000

                 accuracy                           0.28    199960
                macro avg       0.49      0.28      0.28    199960
             weighted avg       0.49      0.28      0.28    199960
```

References:

1: https://sebastianraschka.com/Articles/2014_naive_bayes_1.html

2, https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html

3. https://www.geeksforgeeks.org/python-string-replace/

4. https://www.programiz.com/python-programming/methods/string/lower

5. https://www.programiz.com/python-programming/methods/list/append

6. https://www.w3schools.com/python/python_file_open.asp

7. https://scikit-learn.org/stable/modules/generated/sklearn.metrics.classification_report.html

8. https://geronimo.ai/an-introduction-to-the-naive-bayes-classifier-for-text-classification-why-so-serious/

9. https://towardsdatascience.com/word-bags-vs-word-sequences-for-text-classification-e0222c21d2ec

10. https://www.youtube.com/watch?v=PPeaRc-r1OI&t=655s

11. https://www.youtube.com/watch?v=BqUmKsfSWho