2198-CSE-6363-001 Machine Learning

Project 2 Report

Note: I have kept [my_path =r'C:\Users\Admin\Desktop\20News\20_newsgroups'] Please use appropriate path whilst running the ipython file o your machine. P.S as it is not object oriented, please execute all the kernels in sequence.

The given dataset is a newsgroup of 20 clusters *(http://www.cs.cmu.edu/afs/cs/project/theo-11/www/naive-bayes.html).* It is a program that performs statistical text classification. It is based on the *Bow* library.

Introduction:

The general pattern of rainbow usage is in two steps (1) have rainbow reads the documents and write to disk a "model" containing their statistics, (2) using the model, rainbow performs classification or diagnostics.

We can obtain on-line documentation of each rainbow command-line option by typing

```
 rainbow --help | more
```

This --help option is useful checking the latest details of particular options, but does not provide a tutorial or an overview of rainbow's use.

Command-line options in rainbow and all the *Bow* library frontends are handled by the libargp library from the FSF. Many command-line options have both long and short forms. For example, to set the verbosity level to 4 (to make rainbow give more runtime diagnostic messages than usual), you can type "--verbosity=4", or "--verbosity 4", or "-v 4".

Reading the documents, building a model

Before performing classification or diagnostics with rainbow, you must first have rainbow index your data--that is, read your documents and archive a "model" containing their statistics. The text indexed for the model must contain all the training data. The testing data may also be read as part of the model, or it can be left out and read later.

The model is placed in the file system location indicated by the -d option. If no -d option is given, the name ~/.rainbow is used by default. (The model name is actually a file system directory containing separate files for different aspects of the model. If the model directory location does not exist when rainbow is invoked, rainbow will create it automatically.)

In the most basic setting, the text data should be in plain text files, one file per document. No special tags are needed at the beginning or end of documents. Thus, for example, you should be able to index a directory of UseNet articles or MH mailboxes without any preprocessing. The files should be organized in directories, such that all documents with the same class label are contained within a directory. (Rainbow does not directly support classification tasks in which individual documents have multiple class labels. I recommend handling this as a series of binary classification tasks.)

To build a model, call rainbow with the --index (or -i) option, followed by one directory name for each class.

Tokenizing Options:

When indexing a file, rainbow turns the file's stream of characters into tokens by a process called tokenization or "lexing".

By default, rainbow tokenizes all alphabetic sequences of characters (that is characters in A-Z and a-z), changing each sequence to lowercase and tossing out any token which is on the "stoplist", a list of common words such as "the", "of", "is", etc.

Rainbow supports several options for tokenizing text. For example the --skip-headers (or -h) option causes rainbow to skip newsgroup or email headers before beginning tokenization.

Classifying Documents:

Once indexing is performed and a model has been archived to disk, rainbow can perform document classification. Statistics from a set of *training* documents will determine the parameters of the classifier; classification of a set of *testing* documents will be output.

The --test (or -t) option performs a specified number of trials and prints the classifications of the documents in each trial's test-set to standard output. For example,

```
  rainbow -d ~/model --test-set=0.4 --test=3
```

will output the results of three trials, each with a randomized test-train split in which 60 percent of the documents are used for training, and 40 percent for testing. Details of the --test-set option are described in section 3.1.

That is, one test file per line, consisting of the following fields:

```
  directory/filename TrueClass TopPredictedClass:score1 2ndPredictedClass:score2 ...
```

The Perl script rainbow-stats, which is provided in the Bow source distribution, reads lines like this and outputs average accuracy, standard error, and a confusion matrix.

Selecting the Classification Method

Rainbow supports several different classification methods, (and the code makes it easy to add more). The default is Naive Bayes, but k-nearest neighbor, TFIDF, and probabilistic indexing are all available. These are specified with the --method (or -m) option, followed by one of the following keywords: naivebayes, knn, tfidf, prind.

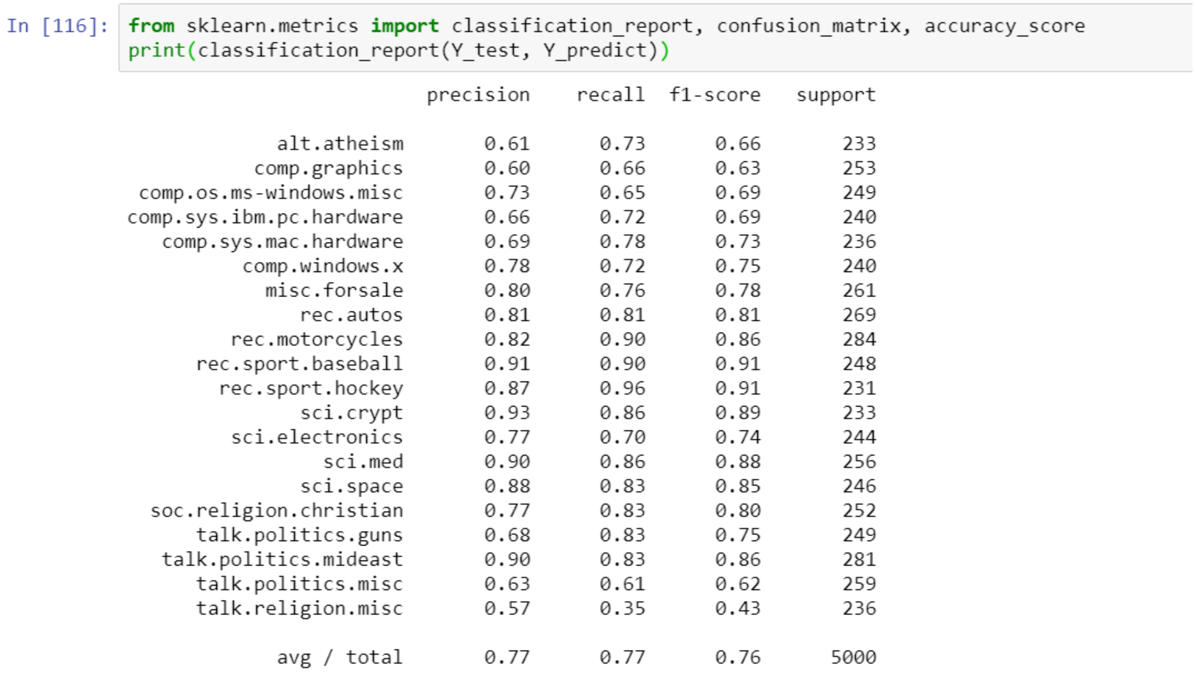
For this project we are using Naive Bayes Option,

The following options change parameters of Naive Bayes.

--smoothing-method=METHOD	Set the method for smoothing word probabilities to avoid zeros; METHOD may be one of: goodturing, laplace, mestimate, wittenbell. The default is laplace, which is a uniform Dirichlet prior with alpha=2.
--event-model=EVENTNAME	Set what objects will be considered the `events' of the probabilistic model. EVENTNAME can be one of: word (i.e. multinomial, unigram), document (i.e. multi-variate Bernoulli, bit vector), or document-then-word (i.e. document-length-normalized multinomial).
--uniform-class-priors	When classifying and calculating mutual information, use equal prior probabilities on classes, instead of using the distribution determined from the training data.

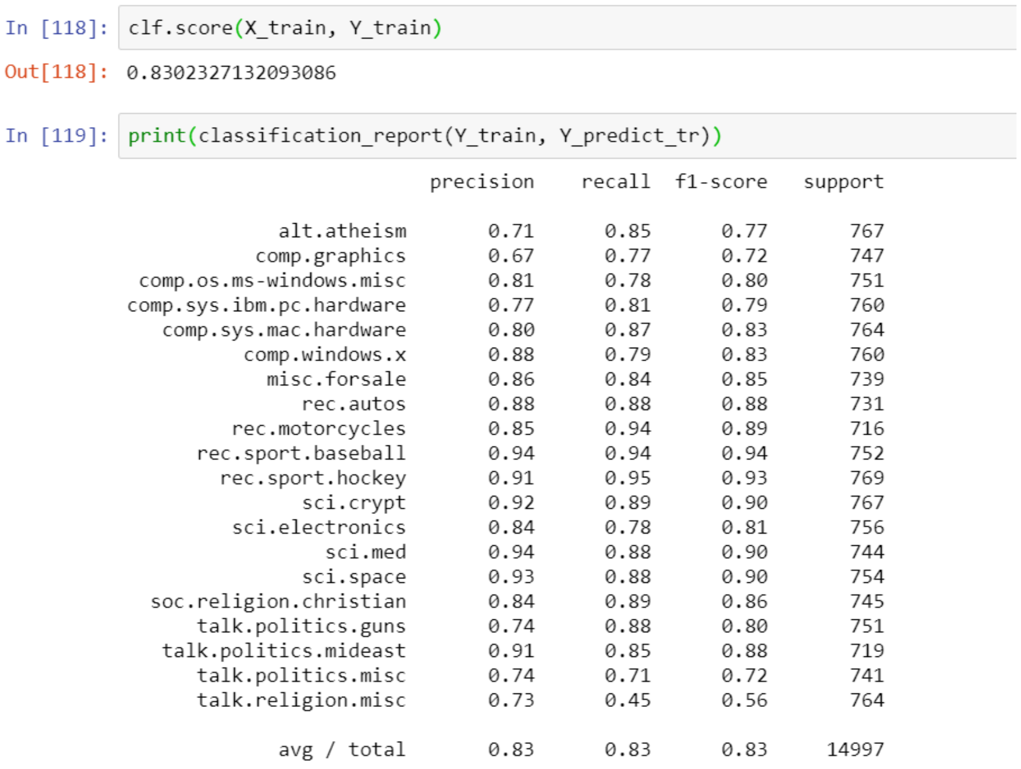
Conclusion:

Classification Reports:

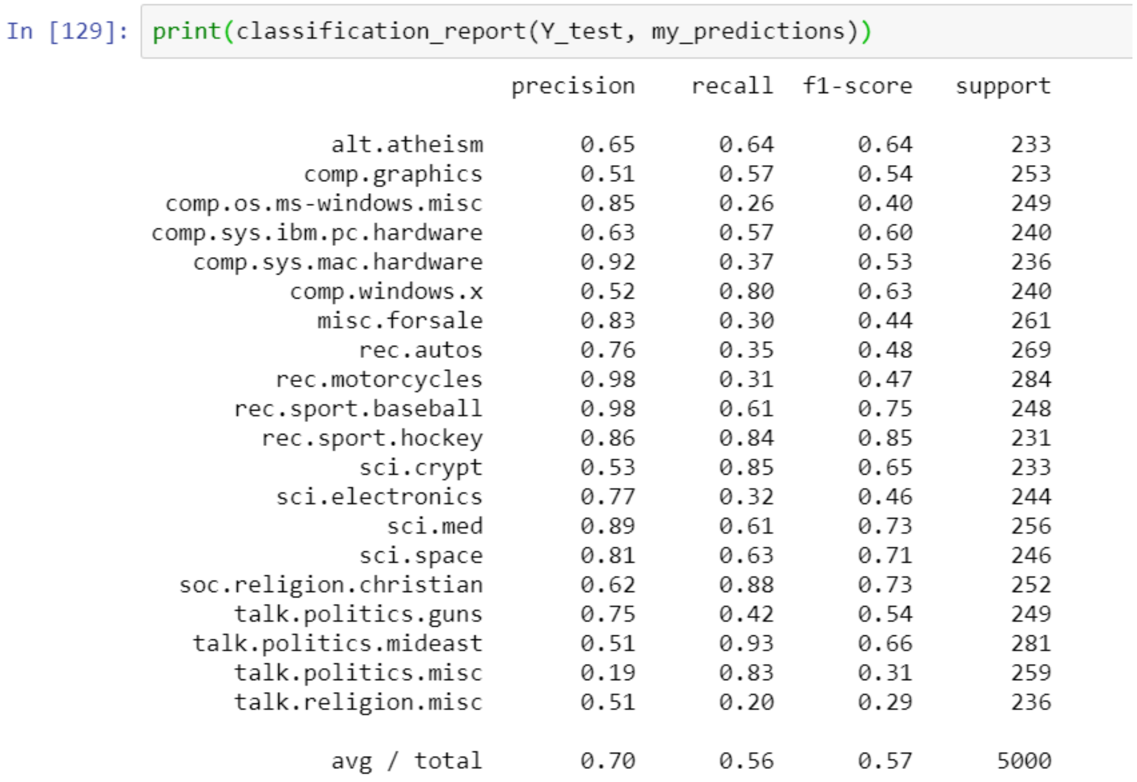


```
In [116]: from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
          print(classification_report(Y_test, Y_predict))
```

```
                          precision    recall  f1-score   support

             alt.atheism       0.61      0.73      0.66       233
           comp.graphics       0.60      0.66      0.63       253
 comp.os.ms-windows.misc       0.73      0.65      0.69       249
comp.sys.ibm.pc.hardware       0.66      0.72      0.69       240
   comp.sys.mac.hardware       0.69      0.78      0.73       236
          comp.windows.x       0.78      0.72      0.75       240
            misc.forsale       0.80      0.76      0.78       261
               rec.autos       0.81      0.81      0.81       269
         rec.motorcycles       0.82      0.90      0.86       284
      rec.sport.baseball       0.91      0.90      0.91       248
        rec.sport.hockey       0.87      0.96      0.91       231
               sci.crypt       0.93      0.86      0.89       233
         sci.electronics       0.77      0.70      0.74       244
                 sci.med       0.90      0.86      0.88       256
               sci.space       0.88      0.83      0.85       246
  soc.religion.christian       0.77      0.83      0.80       252
      talk.politics.guns       0.68      0.83      0.75       249
   talk.politics.mideast       0.90      0.83      0.86       281
      talk.politics.misc       0.63      0.61      0.62       259
      talk.religion.misc       0.57      0.35      0.43       236

             avg / total       0.77      0.77      0.76      5000
```

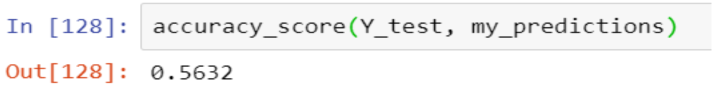


```
In [118]: clf.score(X_train, Y_train)

Out[118]: 0.8302327132093086

In [119]: print(classification_report(Y_train, Y_predict_tr))

                          precision    recall  f1-score   support

             alt.atheism       0.71      0.85      0.77       767
           comp.graphics       0.67      0.77      0.72       747
 comp.os.ms-windows.misc       0.81      0.78      0.80       751
comp.sys.ibm.pc.hardware       0.77      0.81      0.79       760
   comp.sys.mac.hardware       0.80      0.87      0.83       764
          comp.windows.x       0.88      0.79      0.83       760
            misc.forsale       0.86      0.84      0.85       739
               rec.autos       0.88      0.88      0.88       731
         rec.motorcycles       0.85      0.94      0.89       716
      rec.sport.baseball       0.94      0.94      0.94       752
        rec.sport.hockey       0.91      0.95      0.93       769
               sci.crypt       0.92      0.89      0.90       767
         sci.electronics       0.84      0.78      0.81       756
                 sci.med       0.94      0.88      0.90       744
               sci.space       0.93      0.88      0.90       754
  soc.religion.christian       0.84      0.89      0.86       745
      talk.politics.guns       0.74      0.88      0.80       751
   talk.politics.mideast       0.91      0.85      0.88       719
      talk.politics.misc       0.74      0.71      0.72       741
      talk.religion.misc       0.73      0.45      0.56       764

             avg / total       0.83      0.83      0.83     14997
```



```
In [129]: print(classification_report(Y_test, my_predictions))

                          precision    recall  f1-score   support

             alt.atheism       0.65      0.64      0.64       233
           comp.graphics       0.51      0.57      0.54       253
 comp.os.ms-windows.misc       0.85      0.26      0.40       249
comp.sys.ibm.pc.hardware       0.63      0.57      0.60       240
   comp.sys.mac.hardware       0.92      0.37      0.53       236
          comp.windows.x       0.52      0.80      0.63       240
            misc.forsale       0.83      0.30      0.44       261
               rec.autos       0.76      0.35      0.48       269
         rec.motorcycles       0.98      0.31      0.47       284
      rec.sport.baseball       0.98      0.61      0.75       248
        rec.sport.hockey       0.86      0.84      0.85       231
               sci.crypt       0.53      0.85      0.65       233
         sci.electronics       0.77      0.32      0.46       244
                 sci.med       0.89      0.61      0.73       256
               sci.space       0.81      0.63      0.71       246
  soc.religion.christian       0.62      0.88      0.73       252
      talk.politics.guns       0.75      0.42      0.54       249
   talk.politics.mideast       0.51      0.93      0.66       281
      talk.politics.misc       0.19      0.83      0.31       259
      talk.religion.misc       0.51      0.20      0.29       236

             avg / total       0.70      0.56      0.57      5000
```

Accuracy:



```
In [128]: accuracy_score(Y_test, my_predictions)

Out[128]: 0.5632
```