CSE-6363 Assignment 2 Swetha Vijaya Raghavan

1001551229

Bayesian learning for classifying net news text articles i.e., on the 20 newsgroups dataset using half data as training data, and the other half as testing data implemented in python.

```
pathDir = os.listdir(os.getcwd() + '/newsgroups')
fileList = {}
testingSet = {}
trainingSet = {}
for folderName in pathDir:
    appendingPath = os.getcwd() + '/newsgroups/' + folderName + '/'
    pathFile = os.listdir(os.getcwd() + '/newsgroups/' + folderName)
    fileList[folderName] = pathFile
    randomList = range(0, len(pathFile))
    random.shuffle(randomList)
    # 50-50 split for training & testing
    trainingSet[folderName] = list(map(lambda x: pathFile[x], randomList[(len(pathFile) / 2):]))
    testingSet[folderName] = list(map(lambda x: appendingPath + pathFile[x], randomList[:(len(pathFile) / 2)]))
```

Now, we need to read the words, so the below implemented `readingWords` method with file as parameter, reads in lower case which is trimmed using regular expression to get rid of unnecessary special characters.

```
# Regular Expression for words
def readingWords(file):
    with open(file, 'r') as word:
        strings = word.read().lower()
        return re.findall(r"[\w']+", strings)
```

This method `wordProbability` calculates the probability of each word by avoiding zero probability error

```
# probability of the class words by avoiding zero probability error
def wordProbabiity(hash, word, den):
    word = word.lower()
    if word in hash:
        return math.log(hash[word] + 1.0) / den
    return math.log(1.0 / den)
```

by adding 1 to numerator and adding total count to the denominator.

The total probability is calculated using ` readingWords ` & ` wordProbability ` methods.

```
# Total Probability calculation
def totalProbability(file, hash, den):
    li = list(map(lambda x: wordProbabiity(hash, x, den), readingWords(file)))
    return reduce(lambda x, y: x + y, li)
```

Now we have the necessary probabilities to classify the test data by calculating the maximum probability.

```
# classification
def classification(file, hashing, traningSum, counter):
    print(file)
    keys = traningSum.keys()
    probability = list(map(lambda x: totalProbability(file, hashing[x], trainingSum[x] + counter), keys))
    minimumValue = min(probability)
    maximumValue = max(probability)
    median = (maximumValue + minimumValue) / 2
    probability = list(map(lambda x: x - maximumValue, probability))
    den = sum(list(map(lambda x: math.exp(x), probability)))
    probability = list(map(lambda x: math.exp(x) / den, probability))
    maximumValue = max(probability)
    print(maximumValue)
    maximumIndex = [index for index in range(len(probability)) if probability[index] == maximumValue]
    if (len(maximumIndex) > 1):
        print 'Cannot classify as it has the same probability'
    return keys[maximumIndex[0]]
```

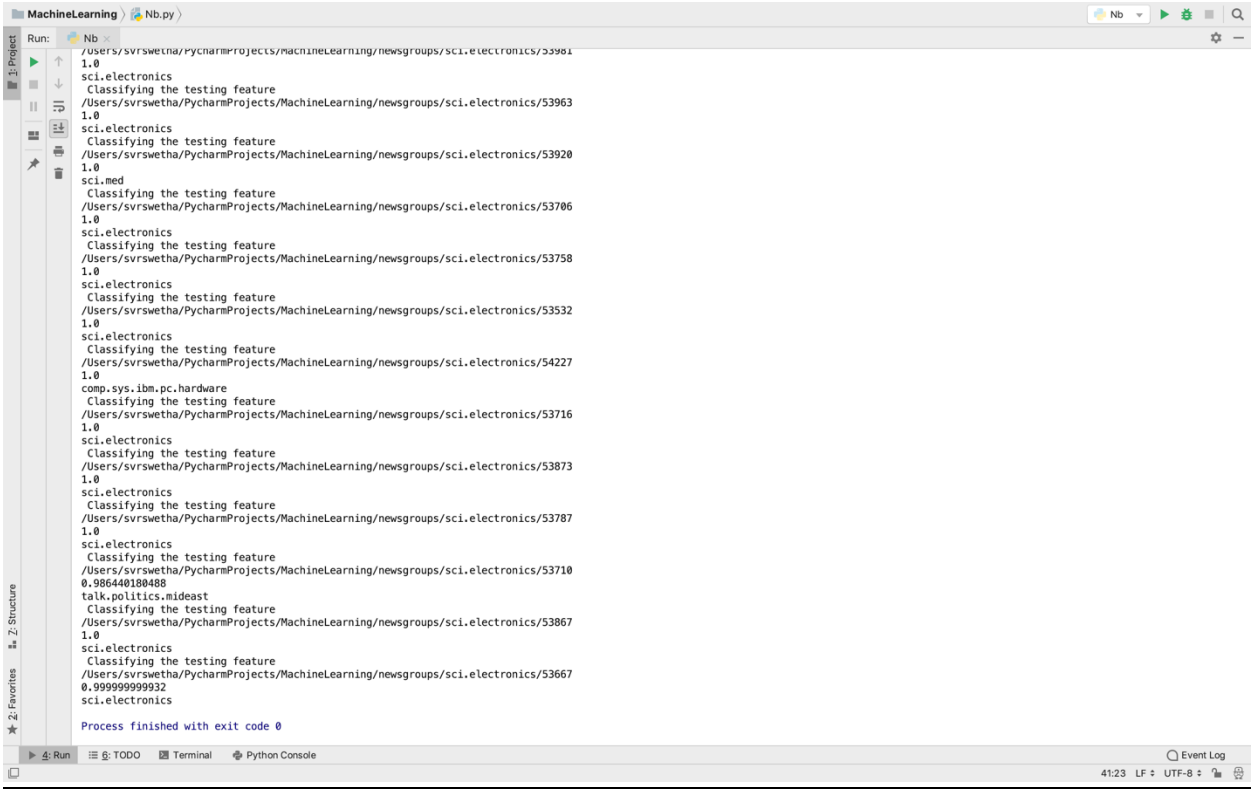
The below code is used to count the unique word occurrences for each document based on each class using built in counter() method.

```
# Unique Word Counting
uniqueWordCount = dict(zip(pathDir, list(map(lambda x: len(fileList[x]), pathDir))))
trainingCount = {}
count = Counter()
trainingSum = {}
for key in pathDir:
    print key
    cwd = os.getcwd() + '/newsgroups/' + key + '/'
    cnt = Counter()
    for fi in trainingSet[key]:
        cnt = cnt + Counter(readingWords(cwd + str(fi)))
    count = count + cnt
    trainingCount[key] = dict(cnt)
    trainingSum[key] = sum(trainingCount[key].values())
count = len(dict(count).keys())
```

The following snippet is used to classify (using naïve bayes) the test data method which has half of the documents from each class using our obtained trained model.

```
# Classifying test data using the trained model
print(testingSet.keys()[0])
length = len(testingSet.keys())
for i in range(0, length):
    fileLength = len(testingSet[testingSet.keys()[i]])
    for j in range(0, fileLength):
        print(" Classifying the testing feature")
        print(classification(testingSet[testingSet.keys()[i]][j], trainingCount, trainingSum, count))
```

Output of the classifier in which we can see the file along with its path that is being tested and also shows the output of the classifier as sci.med, sci.electronics etc.



References:

https://machinelearningmastery.com/naive-bayes-classifier-scratch-python/

https://web.stanford.edu/class/cs124/lec/naivebayes.pdf

https://towardsdatascience.com/multinomial-naive-bayes-classifier-for-text-analysis-python-8dd6825ece67

GitHub & Wikipedia