Assignment 2

Submitted by Maryam Heidari

ALY 6020- Predictive Analytics- CRN: 20770 Northeastern University

Date of Submission: 30 February 2020

Introduction:

In this assignment, we are going to get familiar with Naive Bayes method which is one of the methods of classification for text and character data set. This method is based on probability of events.

Part A:

In the part one, I am going to use the example in the chapter 4 of Machine Learning with R (Lantz) and follow the five steps. In this example I used the data adapted from the SMS Spam which I downloaded from:

https://github.com/stedy/Machine-Learning-with-R-datasets/blob/master/sms_spam.csv

This data set has 2 columns which are type and text. In the type we can see if the message is spam or not, and in the text, we have a content of the message. We are going to use the pattern in texts and predict if the next message is spam or not.

```
 ham spam
4827  747
```

As you can see in above, in this data set 4827 messages are regular and 747 messages are spam. Now, I need to clean the text column to able make a pattern out of it. For that, I install the tm package.

The first step in processing text data involves creating a corpus, which is a collection of text documents. In order to create a corpus, we'll use the VCorpus() function in the tm package, which refers to a volatile corpus.

```
> print(sms_corpus)
<<VCorpus>>
Metadata:  corpus specific: 0, document level (indexed): 0
Content:  documents: 5574
```

After that, we need to clean the text column. In order to that, we need to make all of the words lowercases, remove numbers, remove stop words (which are and, or, etc.), remove punctuations, and last, I am stemming it which means reducing words to their root form and removing the with spaces. I recode the result of all of these steps in sms_corpus_clean.

Now, I need to split the messages into individual components through a process called tokenization. A token is a single element of a text string; in this case, the tokens are words. The result of here is making a matrix that rows indicate documents (SMS messages) and columns indicate terms (words). The result is:

```
> sms_dtm
<<DocumentTermMatrix (documents: 5574, terms: 6592)>>
Non-/sparse entries: 42608/36701200
Sparsity           : 100%
Maximal term length: 40
Weighting          : term frequency (tf)
```

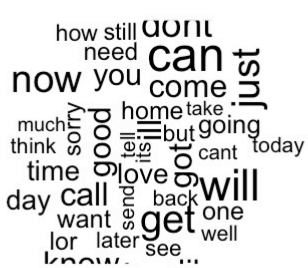
After all of these steps, my data set is ready to split in to test and train data set.

```
> prop.table(table(sms_train_labels))
sms_train_labels
      ham      spam 
0.8647158 0.1352842 
> prop.table(table(sms_test_labels))
sms_test_labels
      ham      spam 
0.8697842 0.1302158 
```

I want to visually depict the frequency at which words appear in text data. One way to do that is word cloud. The cloud is composed of words scattered somewhat randomly around the figure. Words appearing more often in the text are shown in a larger font, while fewer common terms are shown in smaller fonts.

The result for spam is:

And for ham is:



The final step in the data preparation process is to transform the sparse matrix into a data structure that can be used to train a Naive Bayes classifier. For that I need to find frequent words. I used the function that will display the words appearing at least five times in the sms_dtm_train matrix. Then I need to filter our DTM to include only the terms appearing in a specified vector. I want all the rows, but only the columns representing the words in the sms_freq_words vector.

Then, because Naive Bayes classifier is good with categorical data set, we need to convert the matrix to categorical variable that simply indicates yes or no depending on whether the word appears at all instead of the 0 and 1. And when I applied this function on my matrix, I used MARGIN = 2, since we're interested in the columns (MARGIN = 1 is used for rows).

The next step after all of these preparations is training the model, which I did by this line of code: sms_classifier <- naiveBayes(sms_train, sms_train_labels)

Then I need to evaluate my model performance. So, I used the predict () function to make the predictions and then the crosstable () function to compare the result with test data set. And the result is:

```
   Cell Contents
|-------------------------|
|                       N |
|           N / Row Total |
|           N / Col Total |
|-------------------------|


Total Observations in Table:  1390


             | actual
   predicted |       ham |      spam | Row Total |
-------------|-----------|-----------|-----------|
         ham |      1200 |        20 |      1220 |
             |     0.984 |     0.016 |     0.878 |
             |     0.993 |     0.110 |           |
-------------|-----------|-----------|-----------|
        spam |         9 |       161 |       170 |
             |     0.053 |     0.947 |     0.122 |
             |     0.007 |     0.890 |           |
-------------|-----------|-----------|-----------|
Column Total |      1209 |       181 |      1390 |
             |     0.870 |     0.130 |           |
-------------|-----------|-----------|-----------|
```

As you can see, the total of 9+20 = 29 out of 1390 SMS messages were incorrectly classified. Among the errors were 9 out of 1209 ham messages that were misidentified as spam, and 20 of the 181 spam messages were incorrectly labeled as ham. This is a good result. But let's try to improve it. This time, I set laplace = 1 in my model, and the result is:

```
   Cell Contents
|-------------------------|
|                       N |
|           N / Col Total |
|-------------------------|


Total Observations in Table:  1390


             | actual
   predicted |       ham |      spam | Row Total |
-------------|-----------|-----------|-----------|
         ham |      1202 |        28 |      1230 |
             |     0.994 |     0.155 |           |
-------------|-----------|-----------|-----------|
        spam |         7 |       153 |       160 |
             |     0.006 |     0.845 |           |
-------------|-----------|-----------|-----------|
Column Total |      1209 |       181 |      1390 |
             |     0.870 |     0.130 |           |
-------------|-----------|-----------|-----------|
```

This time, the total of 7+28 = 35 out of 1390 SMS messages were incorrectly classified. Among the errors were 7 out of 1209 ham messages that were misidentified as spam, and 28 of the 181 spam messages were incorrectly labeled as ham.

Adding the Laplace estimator reduced the number of false positives from 9 to 7 and the number of false negatives from 20 to 28. It is a small change, but I personally, prefer the previous model because the second one has 6 more mislead.

Part B:

In this part, we need to repeat all of the steps for new data set. I find a data set about the amazon label. They use customers review to see if the product is good or not. I found my data set from Kaggle.com (https://www.kaggle.com/marklvl/sentiment-labelled-sentences-data-set#amazon_cells_labelled.csv). It has 999 observation and 6 variables. I eliminate it to 2 variables and delete those rows which has text instead of 0 and 1 (which shows product is good or not). Now I have data set with 2 variable and 770 observation. And I change the 0 and 1 to great and bad. Also, I change the name of columns to text and type. Now the table of type column is:

```
great   bad
  383   387
```

Then I started cleaning the text column, like the part A. and then split the data set to 75% and 25 % train and test data set. And then, I compared the proportion in the training and test data frames:

```
> prop.table(table(amazon_train_labels))
amazon_train_labels
    great       bad
0.4835355 0.5164645
> prop.table(table(amazon_test_labels))
amazon_test_labels
    great       bad
0.5360825 0.4639175
```

The final step in the data preparation process is to transform the sparse matrix into a data structure that can be used to train a Naive Bayes classifier. For that I need to find frequent words.

I used the function that will display the words appearing at least five times in the amazon_dtm_train matrix. Then I need to filter our DTM to include only the terms appearing in a specified vector. I want all the rows, but only the columns representing the words in the amazon_freq_words vector.

Then, because Naive Bayes classifier is good with categorical data set, we need to convert the matrix to categorical variable that simply indicates yes or no depending on whether the word appears at all instead of the 0 and 1. And when I applied this function on my matrix, I used MARGIN = 2, since we're interested in the columns (MARGIN = 1 is used for rows).

The next step after all of these preparations is training the model, which I did by this line of code: amazon_classifier <- naiveBayes(amazon_train, amazon_train_labels)

Then I need to evaluate my model performance. So, I used the predict () function to make the predictions and then the crosstable () function to compare the result with test data set. And the result is:

```
   Cell Contents
|-------------------------|
|                       N |
|           N / Row Total |
|           N / Col Total |
|-------------------------|


Total Observations in Table:  194


             | actual
   predicted |     great |       bad | Row Total |
-------------|-----------|-----------|-----------|
       great |        73 |        29 |       102 |
             |     0.716 |     0.284 |     0.526 |
             |     0.702 |     0.322 |           |
-------------|-----------|-----------|-----------|
         bad |        31 |        61 |        92 |
             |     0.337 |     0.663 |     0.474 |
             |     0.298 |     0.678 |           |
-------------|-----------|-----------|-----------|
Column Total |       104 |        90 |       194 |
             |     0.536 |     0.464 |           |
-------------|-----------|-----------|-----------|
```

As you can see, the total of 31+29 = 60 out of 194 comments were incorrectly classified. Among the errors were 31 out of 104 bad comments that were misidentified as great, and 29 of the 90 bad messages were incorrectly labeled as great. This is not a good result. But let's try to improve it. This time, I set laplace = 1 in my model, and the result is:

```
   Cell Contents
|-------------------------|
|                       N |
|           N / Col Total |
|-------------------------|


Total Observations in Table:  194


             | actual
   predicted |     great |       bad | Row Total |
-------------|-----------|-----------|-----------|
       great |        72 |        28 |       100 |
             |     0.692 |     0.311 |           |
-------------|-----------|-----------|-----------|
         bad |        32 |        62 |        94 |
             |     0.308 |     0.689 |           |
-------------|-----------|-----------|-----------|
Column Total |       104 |        90 |       194 |
             |     0.536 |     0.464 |           |
-------------|-----------|-----------|-----------|
```

This time, the total of 32+28 = 60 out of 194 comments were incorrectly classified. Among the errors were 32 out of 104 great comments that were misidentified as bad, and 28 of the 90 bad messages were incorrectly labeled as great.

Adding the Laplace estimator increase the number of false positives from 31 to 32 and reduce the number of false negatives from 29 to 28. It is a small change, and the total error is the same. So it does not have that much affect.

Conclusion:

I learned about the Naive Bayes method, how it , how we clean a text data set and make it ready to make a model based on that, and how I can improve the model.

References:

Ch. 3 of Machine Learning with R (Lantz), in pp. 75-87

https://www.kaggle.com/marklvl/sentiment-labelled-sentences-data-set#amazon_cells_labelled.csv