ALY 6020 ASSIGNMENT 2

RUHIKA FERNANDES

NORTHEASTERN UNIVERSITY

COLLEGE OF PROFESSIONAL STUDIES -MPS ANALYTICS

PROFESSOR : JUSTIN GROSZ

DATE:09/29/2019

:

INTRODUCTION

The Naive Bayes classifier is a family of algorithms which are built on the principle of Bayes theorem which demonstrates conditional probability. Bayes theorem states that if A and B are two dependent events the best estimate of P(A|B) is the proportion of trials in which A occurred with B out of all the trials in which B occurred. In plain language, this tells us that if we know event B occurred, the probability of event A is higher more often that A and B occur together each time B is observed. Therefore the probability is observed as $P(A|B)=P(A\cap B)/P(B)=(P(B|A)*P(A))/P(B)$ where P(A) is the prior probability , P(A|B) is the posterior probability , P(B|A) is the likelihood probability and P(B) is marginal likelihood .Naive Bayes makes an assumption that each feature in a dataset makes an independent and equal contribution to the outcome . It is one of the most common machine learning techniques that makes use of Bayesian methods . It is simple fast and effective and performs well with noisy data as well as missing data . However it relies on the faulty assumption that all the features are independent and equally important to the outcome. Because probabilities in the Naive Bayes formula are multiplied in a chain, some time a 0% chance of occurrence causes percent value causes the posterior probability to become 0 as well . A solution to this problem involves using something called the Laplace estimator, which is named after the French mathematician Pierre-Simon Laplace. The Laplace estimator essentially adds a small number to each of the counts in the frequency table, which ensures that each feature has a nonzero probability of occurring with each class. Let us observe the implementation of the Naive Bayes classification through its implementation in R as part of this assignment .

ANALYSIS

PART A

Step1:Collect and store the data

Data is obtained from http://www.dt.fee.unicamp.br/~tiago/smsspamcollection/ that has a dataset that contains messages and a column named type indicating If the message is spam or legitimate . The Junk messages are termed as “spam”and the legitimate ones as ”ham”. The dataset used can be found in the Appendix

Step2:Explore and prepare the data

In this case the data is first stored into a dataframe sms_ds. After that it is explored using the str command from here we can observe that there are 5574 and the two features which are the messages and the type

```
> str(sms_ds)
'data.frame':   5574 obs. of  2 variables:
 $ type: chr  "ham" "ham" "spam" "ham" ...
 $ text: chr  "Go until jurong point, crazy.. Available only in bugis n great world la e buffet... Cine there got am
ore wat..." "Ok lar... Joking wif u oni..." "Free entry in 2 a wkly comp to win FA Cup final tkts 21st May 2005. Tex
t FA to 87121 to receive entry question("| __truncated__ "U dun say so early hor... U c already then say..." ...
>
```

It can be observed that the type variable is a categorical variable and hence it should be converted to a factor , for that purpose the factor method is used to do so giving it two levels “spam” and “ham”. Examining it with str () and table () function that shows us how many entries come under the spam and ham category respectively we hence know that there are 4827 enteries of ham and 747 spam enteries. Since the data is text data it needs to be cleaned and standardized and for that purpose words , spaces, numbers and punctuation needs to be remove and hence we use the text mining package tm . In order to install tm we use install.packages (tm) and library (tm) to load it respectively . Since the data set contains text messages we will need a corpus to store that data and hence we will use VCorpus to store it on memory as opposed to data as this is a volatile storage

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

The corpus is a complex list and hence various list fucntions can be used to acess the contents of the corpus .The inspect command and thes as.character command are used to observe the contents of the corpus .In order to observe specific summary of first 3 messages in the corpus we use inspect command

```
> inspect(sms_corpus[1:3])
<<VCorpus>>
Metadata:  corpus specific: 0, document level (indexed): 0
Content:  documents: 3

[[1]]
<<PlainTextDocument>>
Metadata:  7
Content:  chars: 111

[[2]]
<<PlainTextDocument>>
Metadata:  7
Content:  chars: 29

[[3]]
<<PlainTextDocument>>
Metadata:  7
Content:  chars: 155
```

in order to observe content we use as.character.First we need to standardize all the components of the corpus so that same upper case and lowercase words are treated as the same word for this pupose we use tm wrapper fucntion content transformer() in order to treat the tolower funtion as a tranformation to the corpus .the fucntion tm_map () permits for transformations on the corpus .After that we use the removeNumbers() as number don't form a pattern , removeWords()which come under filler word category which are stop words() , stripWhitespace() to remove whitespaces and removePunctuations() in order to clean the corpus further . However unlike tolower() these functions do not need the content transformer as these fucntions do need to be wrapped .Following that is required that each word of the text remain in its root form and for that purpose we use the SnowballC package that has wordstem() that does the needful . In order to apply wordstem() to the entire corpus it is required that we use stemdocument to do so. We have managed to clean the text and now the cleaned corus is stored as sms_corpus_clean. In order to inspect it we use the print command and notice the state of the cleaned corpus ,we found out that the data is fairly clean as both the results are the same

BEFORE

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

AFTER

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

Post that the DocumentTermMatrix () is used to create a frequency table with words that are present in the clean corpus . It should be noted that the transformation applied above can also be applied while using the DocumentTermMatrix () fucntion with the help of the control paramater applying all the list fucntions

```
> sms_dtm
<<DocumentTermMatrix (documents: 5574, terms: 6611)>>
Non-/sparse entries: 42653/36807061
Sparsity           : 100%
Maximal term length: 40
Weighting          : term frequency (tf)
> sms_dtm2
<<DocumentTermMatrix (documents: 5574, terms: 7005)>>
Non-/sparse entries: 43742/39002128
Sparsity           : 100%
Maximal term length: 40
Weighting          : term frequency (tf)
>
```

We observe that their contents differ as in the second case the clean up only occurs when the contents of the corpus are aldready split into words by the DocumentTermMatrix() . Because of that the processing happenes on the split up words rather than happening on the corpus which is having all the messgaes together . This is not very significant in this case ad hence the difference in the values can be ignored .However in case the values varied singificantly it would be very important to choose which of the methods were better

We proceed to create the train and test dataset . The prop.table() fucntion shows us the proportion of spam and ham in our test and training dataset . It is noticed that both train and test dataset have about 13% spam

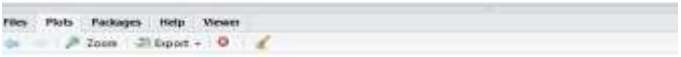
```
> prop.table(table(sms_train_labels))
sms_train_labels
      ham      spam
0.8647158 0.1352842
> prop.table(table(sms_test_labels))
sms_test_labels
      ham      spam
0.8697842 0.1302158
>
```

We need to visualize the data and hence for that purpose we use word clouds . In order to do that we need to install and load the wordcloud package in R . We can directly use the word cloud fucntion on out created corpus and we get the following visualization as we have a random order

we can get separate word clouds for spam and ham respectively

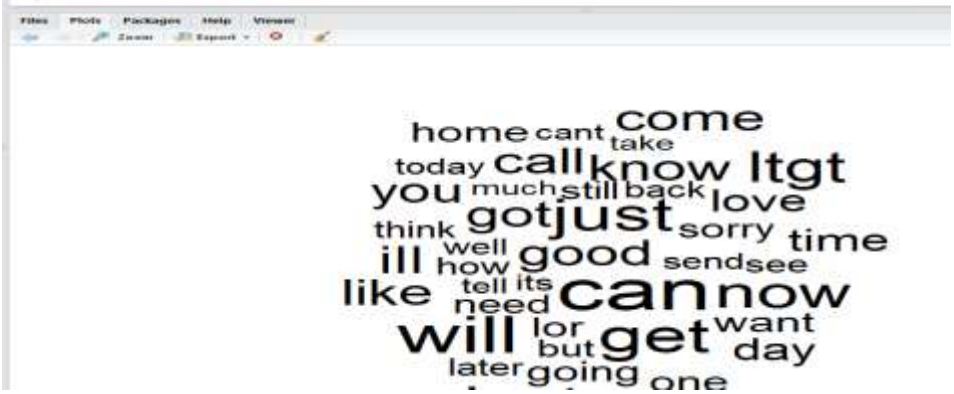
Spam word cloud as observed below:

```
> wordcloud(spam$text, max.words = 40, scale = c(5, 0.5))
Warning messages:
1: In tm_map.SimpleCorpus(corpus, tm::removePunctuation) :
  transformation drops documents
2: In tm_map.SimpleCorpus(corpus, function(x) tm::removeWords(x, tm::stopwords())) :
  transformation drops documents
>
```



Ham word cloud as observed below

```
> wordcloud(ham$text, max.words = 40, scale = c(5, 0.5))
Warning messages:
1: In doTryCatch(return(expr), name, parentenv, handler) :
  invalid input 'ÀÀ1000' in 'utf8towcs'
2: In doTryCatch(return(expr), name, parentenv, handler) :
  invalid input 'ÀÀ1000' in 'utf8towcs'
3: In tm_map.SimpleCorpus(corpus, tm::removePunctuation) :
  transformation drops documents
4: In tm_map.SimpleCorpus(corpus, function(x) tm::removeWords(x, tm::stopwords())) :
  transformation drops documents
>
```



As the last step of the data preparation we need to find ou the frequent terms of our DTM and use the the findFrequentTerms fucntion .On keeping the limit as at least 5 messages we undersand that there are 1159 words appearing frequenly in at least 5 messages . Post that DTM is filtered to get words only in a specified vector format and post that since we have to use the Naive Bayes classifier that run on categorical data we proceed to create a convert_count function to obtain the data in the form of a sting in the form of “yes” or “no”. We use the margin=2 in the apply function on the word frequency set to test and train as we are interested in the columns

Step3: Training the model on the data

In order to implement the Naive bayes classifier it is required that we use the e1071 package in R . Create the naïve bayes classifier by using the sms_train set and the sms_train_labels

Step:4 evaluate performance

We use the predict() function to make the predictions . Post that we use the gmodels package in order to create a cross table as well as make us e of the dnn(Dimension names) parameter in order to rename the row and columns. This model shows us that the accuracy is extremely high almost 97% . The false positives are only 9 and the false negatives are 20. This can cause people to miss out on data as 9 ham messages were classified as spam and hence in order to mitigate that we try to improve the model performance

```
> CrossTable(sms_test_pred, sms_test_labels,prop.chisq = FALSE, prop.t = FALSE,dnn = c('predicted', 'actual'))

   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 |           |
-------------|-----------|-----------|-----------|
```

Step 5: Improving the performance

In this case we add a Laplace estimator in order to observe that words that came in 0 spam or ham messages have an undisputed say in the data set and for that it is required that we set the Laplace parameter to one on the same model before we can observe that the number of false positives are reduced to 7 still maintaining the 97% accuracy although the false negatives rose from 20 to 28 so in this case the model without Laplace estimator seems to be better as the total number of incorrectly classified messages have increased

```
> sms_classifier2 <- naiveBayes(sms_train, sms_train_labels,laplace = 1)
> sms_test_pred2 <- predict(sms_classifier2, sms_test)
> CrossTable(sms_test_pred2, sms_test_labels,prop.chisq = FALSE, prop.t = FALSE, prop.r = FALSE,dnn = c('
  'actual'))


   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 |           |
-------------|-----------|-----------|-----------|
```

Summary of learning : Form this exercise we learned that Naïve Bayes works very well with the categorical classification and works well when the number of numerical attributes are low. It is Naïve and makes use of Bayes theorem for conditional probability treating all attributes as independent . In this case in order to enhance its performance Laplace estimator is used however even though it decreased false positives it increased the false negatives by 8 counts thereby increasing error in classification . With the help of this experiment the use of the tm package and the ability to handle text data with the help of the naïve bayes classification was also explored through word cloud and TDM

PART B

Step1:load the data .

In this case we are going to use the Pima Indian dataset present in the appendix . It tells us various factors which are responsible to contribute to diabetes . Based on those features a record is marked as diabetic or nondiabetic .The dataset is retrieved from the site mentioned below https://www.kaggle.com/uciml/pima-indians-diabetes-database

Step2:explore and clean the loaded data

On using the str() function on the dataframe created diabetes_data we find that the number of observations are 768 and the number of columns are 9 we also find out that the number of diabetic and non diabetic entries with the help of the table() function .We also observe that the proportion of diabetic entries is 268 and the nondiabetic entries is 500

```
> str(diabetes_data)
'data.frame':   768 obs. of  9 variables:
 $ Pregnancies             : int  6 1 8 1 0 5 3 10 2 8 ...
 $ Glucose                 : int  148 85 183 89 137 116 78 115 197 125 ...
 $ BloodPressure           : int  72 66 64 66 40 74 50 0 70 96 ...
 $ SkinThickness           : int  35 29 0 23 35 0 32 0 45 0 ...
 $ Insulin                 : int  0 0 0 94 168 0 88 0 543 0 ...
 $ BMI                     : num  33.6 26.6 23.3 28.1 43.1 25.6 31 35.3 30.5 0 ...
 $ DiabetesPedigreeFunction: num  0.627 0.351 0.672 0.167 2.288 ...
 $ Age                     : int  50 31 32 21 33 30 26 29 53 54 ...
 $ Outcome                 : Factor w/ 2 levels "D","ND": 1 2 1 2 1 2 1 2 1 1 ...
>
>
> table(diabetes_data$Outcome)

  D  ND
268 500
```

On careful inspection of the data we can actually observe that some values such as blood pressure ,glucose, BMI , insulin and skin thickness have values which are 0 .But this is not possible and can resulting in proving skewed and improper results because of which we will first convert all the 0 values to NA . From the below screenshot we can observe that all the 0 values have been transformed to NA .

```
> str(diabetes_data)
'data.frame':   768 obs. of  9 variables:
 $ Pregnancies             : int  6 1 8 1 0 5 3 10 2 8 ...
 $ Glucose                 : int  148 85 183 89 137 116 78 115 197 125 ...
 $ BloodPressure           : int  72 66 64 66 40 74 50 NA 70 96 ...
 $ SkinThickness           : int  35 29 NA 23 35 NA 32 NA 45 NA ...
 $ Insulin                 : int  NA NA NA 94 168 NA 88 NA 543 NA ...
 $ BMI                     : num  33.6 26.6 23.3 28.1 43.1 25.6 31 35.3 30.5 NA ...
 $ DiabetesPedigreeFunction: num  0.627 0.351 0.672 0.167 2.288 ...
 $ Age                     : int  50 31 32 21 33 30 26 29 53 54 ...
 $ Outcome                 : Factor w/ 2 levels "D","ND": 1 2 1 2 1 2 1 2 1 1 ...
>
```

However we need to figure out what values Can in place of the 0 and hence we will use the mice package . It creates various predictions regarding the missing data and provides various replacements for the multivariate missing data. The install.packages(mice) and library(mice) are the commands that need to be used for using the mice package . The randomForest package also needs to be installed with mice and library(randomForest) must be used in order to load it . The mice function can then be used taking the diabetes_data as the input .The complete function takes the data and returns it in a specified format. The values generated from mice has then to be placed back into the dataset diabetes_data for the columns having missing values

```
> str(diabetes_data)
'data.frame':   768 obs. of  9 variables:
 $ Pregnancies             : int  6 1 8 1 0 5 3 10 2 8 ...
 $ Glucose                 : int  148 85 183 89 137 116 78 115 197 125 ...
 $ BloodPressure           : int  72 66 64 66 40 74 50 68 70 96 ...
 $ SkinThickness           : int  35 29 30 23 35 17 32 18 45 31 ...
 $ Insulin                 : int  126 115 120 94 168 105 88 145 543 155 ...
 $ BMI                     : num  33.6 26.6 23.3 28.1 43.1 25.6 31 35.3 30.5 22.4 ...
 $ DiabetesPedigreeFunction: num  0.627 0.351 0.672 0.167 2.288 ...
 $ Age                     : int  50 31 32 21 33 30 26 29 53 54 ...
 $ Outcome                 : Factor w/ 2 levels "D","ND": 1 2 1 2 1 2 1 2 1 1 ...
>
```

The revamped structure of the dataframe shows that NA and 0 values have been replaced. The outcome of the data set is a categorical variable and hence needs to be converted using the factor function . It is observed that the outcome table has labels as diabetic and nondiabetic

```
> print(diabetes_data$Outcome)
  [1] diabetic     non_diabetic diabetic     non_diabetic diabetic     non_diabetic diabetic     non_diabet
  [9] diabetic     diabetic     non_diabetic diabetic     non_diabetic diabetic     diabetic     diabetic
 [17] diabetic     diabetic     non_diabetic diabetic     non_diabetic non_diabetic diabetic     diabetic
 [25] diabetic     diabetic     diabetic     non_diabetic non_diabetic non_diabetic non_diabetic diabetic
 [33] non_diabetic non_diabetic non_diabetic non_diabetic non_diabetic diabetic     diabetic     diabetic
 [41] non_diabetic non_diabetic non_diabetic diabetic     non_diabetic diabetic     non_diabetic non_diabet
```

further on using the table () we come to know that 65% of the enteries indicate nondiabetic enteries and the remaining 35% indicate diabetic enteries

```
> round(prop.table(table(diabetes_data$Outcome)) * 100, digits = 1)

non_diabetic     diabetic
        65.1         34.9
>
```

The objective is to improve predictive accuracy and not allow a particular feature impact the prediction due to large numeric value range hence the normaization of data is required .For that purpose hence the normalize fucntion is created having a (value – its minimum value)/range for each point in the dataset . The lappy fucntion ensures that this is applied to the full dataframe for all the columns . We then proceed to create the train and test dataset .The train and test labels are also created. We then proceed to to find out the proportions of diabetic and non diabetic entries in the train and test data respectively . Both the train and test dataset contain about 35% diabetic enteries

```
diabetes_train_labels
non_diabetic     diabetic
   0.6530945    0.3469055
> prop.table(table(diabetes_train_labels))
diabetes_train_labels
non_diabetic     diabetic
   0.6530945    0.3469055
>
```

Step3: Build the classifier and train the model on the data

We need to use the package e1071 for that we will have to use the command install.packages(e1071) and in case it is already installed use the function library(e1071) in order to use the Naïve Bayes classifier . Make use of naiveBayes () method to build the classifier with the input as the training data .

Step4:Evlaue the model performance

With the help of the predict function having inputs as the classifier and test data make the predictions and after that load the package gmodels with the library(gmodels) function . This will enable us to use the CrossTable () in order to create a confusion matrix . During the creation of a confusion matrix make sure to use the dnn parameter to set the actual and predicted values . The matrix tell us that the number of correct classifications are about 75% . whereas the number of false negatives are 22 and false positives are 16 hence 25% erroneous classification

```
> CrossTable(diabetes_test_pred, diabetes_test_labels,prop.chisq = FALSE, prop.t = FALSE,dnn = c('predicte
l'))

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

Total Observations in Table:  154

             | actual
   predicted | non_diabetic |     diabetic |    Row Total |
-------------|--------------|--------------|--------------|
non_diabetic |           83 |           22 |          105 |
             |        0.790 |        0.210 |        0.682 |
             |        0.838 |        0.400 |              |
-------------|--------------|--------------|--------------|
    diabetic |           16 |           33 |           49 |
             |        0.327 |        0.673 |        0.318 |
             |        0.162 |        0.600 |              |
-------------|--------------|--------------|--------------|
Column Total |           99 |           55 |          154 |
             |        0.643 |        0.357 |              |
-------------|--------------|--------------|--------------|
```

Step5: Improving model performance

In order improve model performance and increase the efficiency of the algorithm let us make use of the Z score standardization as there are no predefined minimum and maximum values and the extreme values are not compressed towards the center . Also we will introduced the laplace estimator in order to see if the classification of false positives reduce . We use the scale () in order to carry out the Z score standardization and then again proceed to create a train and test set with that data along with using CrossTable() to generate a confusion matrix . The output indicates that the output is the same as before and there no change in efficiency

```
             | actual
   predicted | non_diabetic |     diabetic |    Row Total |
-------------|--------------|--------------|--------------|
non_diabetic |           83 |           22 |          105 |
             |        0.790 |        0.210 |        0.682 |
             |        0.838 |        0.400 |              |
-------------|--------------|--------------|--------------|
    diabetic |           16 |           33 |           49 |
             |        0.327 |        0.673 |        0.318 |
             |        0.162 |        0.600 |              |
-------------|--------------|--------------|--------------|
Column Total |           99 |           55 |          154 |
             |        0.643 |        0.357 |              |
-------------|--------------|--------------|--------------|
```

Hence we make use of laplace estimator in order to observe if there is a difference . We set the parameter to 1 while constructing the Naïve Bayes classifier . Again while creating the confusion matrix we are getting the same output showing us that this accuracy of 75% is the maximum possible accuracy . With 75% correctly classified and 25% incorrectly classified . Nor are the values of false positives decreasing with the use of the Laplace estimator

	actual		
predicted	non_diabetic	diabetic	Row Total
non_diabetic	83	22	105
	0.790	0.210	0.682
	0.838	0.400	
diabetic	16	33	49
	0.327	0.673	0.318
	0.162	0.600	
Column Total	99	55	154
	0.643	0.357	

Summary of learning :

Through the completion of **PART B** of this assignment we understand that Naïve Bayes classification is not very appropriate for dataset having multiple numerical attributes . Its performance in part A with the spam dataset showed way more efficiency of 97% than the diabetes dataset with only 75% accuracy . In this case of the PIMA Indian dataset even though the Laplace estimator was used to enhance the results nothing happened . Also having a large number of false negatives in the case of the diabetes dataset is really poor performance and also dangerous . This can conclude that Naïve Bayes may not be ideal under any circumstance for evaluating the diabetes dataset .

REFERENCES

Lantz, B. (2015). Machine learning with R: discover how to build machine learning algorithms, prepare data, and dig deep into data prediction techniques with R. Birmingham: Packt Publishing.

Pima Indians Diabetes Database - dataset by data-society. (2016, December 13). Retrieved from https://data.world/data-society/pima-indians-diabetes-database.

APPENDEX


sms_spam.csv


diabetes.csv