Hannah Roach
5-12-2019

Lab Exercise 8: Naïve Bayesian Classifier

Purpose:	This lab is designed to investigate and practice the Naïve Bayesian Classifier analytic technique. After completing the tasks in this lab you should be able to: • Use R functions for Naïve Bayesian Classification • Apply the requirements for generating appropriate training data • Validate the effectiveness of the Naïve Bayesian Classifier with the big data

Tasks:	Tasks you will complete in this lab include: • Use R –Studio environment to code the Naïve Bayesian Classifier • Use the ODBC connection to the "census" database to create a training data set for Naïve Bayesian Classifier from the big data • Use the Naïve Bayesian Classifier program and evaluate how well it predicts the results using the training data and then compare the results with original data

Part 1 – Building Naïve Bayesian Classifier

Workflow Overview

1. Set working directory and review training and test data
2. Install and load library "e1071"
3. Read in and review data
4. Build the Naïve Bayesian classifier Model from First Principles
5. Predict the Results
6. Execute the Naïve Bayesian Classifier with e1071 package
7. Predict the Outcome of "Enrolls" with the Testdata
8. Review results

LAB Instructions

Step	Action
1	Log in with GPADMIN credentials on to R-Studio.

2	**<u>Set working directory and review training and test data</u>**
	1. Set the working directory using the following command:
	`> setwd("~/LAB08")`
	
	• The **“sample1.csv”** file in this directory represents the data worked with in the instructor led training session. The file has a header row, followed by 14 rows of training data.
	• The **testing data** on which you will predict the results should be appended after the **training data**. The data set should read:
	
	`Age,Income,Jobstaisfaction,Desire,Enrolls ←-------Header`
	`<=30,High,No,Fair,No`
	`<=30,High,No,Excellent,No`
	`31 to 40,High,No,Fair,Yes`
	`>40,Medium,No,Fair,Yes`
	`>40,Low,Yes,Fair,Yes`
	`>40,Low,Yes,Excellent,No`
	`31 to 40,Low,Yes,Excellent,Yes`
	`<=30,Medium,No,Fair,No`
	`<=30,Low,Yes,Fair,Yes`
	`>40,Medium,Yes,Fair,Yes`
	`<=30,Medium,Yes,Excellent,Yes`
	`31 to 40,Medium,No,Excellent,Yes`
	`31 to 40,High,Yes,Fair,Yes`
	`>40,Medium,No,Excellent,No`
	`<=30,Medium,Yes,Fair, ←---------testing data`

3	**<u>Install and load library “e1071”</u>**
	Execute the following command to install the required packages and load the libraries:
	`> install.packages("e1071")`
	`> library("e1071")`

Step	Action
4	**Read in and review data**

1. Execute the following to read in the data.

```
> # read the data into a table from the file
> sample <- read.table("sample1.csv",header=TRUE,sep=",")
> # we will now define the data frames to use the NB
classifier
> # we will now define the data frames to use the NB
classifier
> traindata <- as.data.frame(sample[1:14,])
> testdata <- as.data.frame(sample[15,])
```

You now have two data frame objects "**traindata**" and "**testdata**" for running the NB Classifier.

2. Execute the following command to display the data frames, to ensure they are loaded properly.

```
> #Display data frames
> traindata
> testdata
```

3. Screenshot the output for traindata and test data

```
> testdata
    Age Income Jobsatisfaction Desire Enrolls
15 <=30 Medium             Yes   Fair
> traindata
        Age Income Jobsatisfaction    Desire Enrolls
1      <=30   High              No      Fair      No
2      <=30   High              No Excellent      No
3  31 to 40   High              No      Fair     Yes
4       >40 Medium              No      Fair     Yes
5       >40    Low             Yes      Fair     Yes
6       >40    Low             Yes Excellent      No
7  31 to 40    Low             Yes Excellent     Yes
8      <=30 Medium              No      Fair      No
9      <=30    Low             Yes      Fair     Yes
10      >40 Medium             Yes      Fair     Yes
11     <=30 Medium             Yes Excellent     Yes
12 31 to 40 Medium              No Excellent     Yes
13 31 to 40   High             Yes      Fair     Yes
14      >40 Medium              No Excellent      No
```

5

<u>Build the Naïve Bayesian classifier Model from First Principles:</u>

1. The first step in building the model is the computation of prior probabilities. The independent variables here are the “Age”, “Income”, “Jobsatisfaction” and “Desire”. The dependent variable is “Enrolls”
   Compute the prior probabilities of enrollment, P(no), P(yes) first, the counts :

```
> tprior <- table(traindata$Enrolls)
```

then, normalize over the total number of instances to get the probabilities

```
> tprior <- tprior/sum(tprior)
> tprior
```

Screenshots the results of prior probabilities

```
> tprior

          No   Yes
0.000 0.357 0.643
```

2. Compute the summaries that you need to create a Bayes model: P(A|b), b={no, yes}
   First, count up "no" and "yes" by Age:

```
> ageCounts <-table(traindata[,c("Enrolls", "Age")])
```

3. Then, normalize by the total number of "no" and "yes" each to get the conditional probabilities

```
> ageCounts <- ageCounts/rowSums(ageCounts)
```

Display the results on the console and review the conditional probabilities

```
> ageCounts
```

Screenshot the results

```
> ageCounts
       Age
Enrolls  <=30 31 to 40   >40

     No  0.600    0.000 0.400
     Yes 0.222    0.444 0.333
```

4. Do the same for the other variables.

```
> incomeCounts <- table(traindata[,c("Enrolls",
"Income")])
> incomeCounts <- incomeCounts/rowSums(incomeCounts)
>incomeCounts
```

Screenshot

Step	Action

	<pre>> incomeCounts
	        Income
	Enrolls High Low Medium
	
	    No 0.400 0.200 0.400
	    Yes 0.222 0.333 0.444</pre> **> jsCounts <- table(traindata[,c("Enrolls", "Jobsatisfaction")])**
	> jsCounts<-jsCounts/rowSums(jsCounts)
	>jsCounts
	Screenshot<pre>> jsCounts
	        Jobsatisfaction
	Enrolls No Yes
	
	    No 0.800 0.200
	    Yes 0.333 0.667</pre> **> desireCounts <- table(traindata[,c("Enrolls", "Desire")])**
	> desireCounts <- desireCounts/rowSums(desireCounts)
	>desireCounts
	Screenshot<pre>> desireCounts
	        Desire
	Enrolls Excellent Fair
	
	    No 0.600 0.400
	    Yes 0.333 0.667</pre>

Step	Action
6	**Predict the Results:**

1. Use the Naïve Bayesian Classifier formula to compute product of P(A|b), for b={no, yes}. The maximum of the two is the "predicted" result of the dependent variable. In the test data we need to predict the "Enrolls" given the for Age<=30, Income = Medium, Jobsatisfaction = yes and Desire = Fair

```
> pyes <-
          ageCounts["Yes","<=30"]*
          incomeCounts["Yes","Medium"]*
          jsCounts["Yes","Yes"]*
          desireCounts["Yes","Fair"]*
          tprior["Yes"]
```

   followed by

```
> pno <-
          ageCounts["No","<=30"]*
          incomeCounts["No","Medium"]*
          jsCounts["No","Yes"]*
          desireCounts["No","Fair"]*
          tprior["No"]
```

2. The prediction will be max(pyes,pno).

```
> pyes
> pno
> max(pyes,pno)
```

3. What is the predicted result for "Enrolls" for someone's age less than 30, income medium ,JobSatisfaction yes, and desidre Fiar?

```
> print (pyes)
   Yes
0.0282
> print (pno)
     No
0.00686
> print(max(pyes,pno))
[1] 0.0282
```

7	**Execute the Naïve Bayesian Classifier with e1071 package:**
	
	The Naïve Bayes function computes the conditional a-posterior probabilities of a categorical class variable given independent categorical predictor variables using the Bayes rule. The usage takes the form of naiveBayes(formula, data,…) where the arguments are defined as follows:
	
	• **formula** A formula of the form class ~ x1 + x2 + ... Interactions are not allowed.
	• **data** Either a data frame of factors or a contingency table.
	
	• You are modeling for attribute " Enrolls".
	
	1. Use the following commands to execute the model and display the results.
	`> # use the NB classifier`
	`> model <- naiveBayes(Enrolls ~.,traindata)`
	`> # display model`
	`> model`
	
	2. SCREENSHOT THE RESULTS and compare these results to the **apriori probabilities** you manually computed earlier in step 5. Are they the same or different ?
	They are different

```
> model

Naive Bayes Classifier for Discrete Predictors

Call:
naiveBayes.default(x = X, y = Y, laplace = laplace)

A-priori probabilities:
Y
              No   Yes
0.000 0.357 0.643

Conditional probabilities:
     Age
Y       <=30 31 to 40   >40

  No  0.600    0.000 0.400
  Yes 0.222    0.444 0.333

     Income
Y       High   Low Medium

  No  0.400 0.200  0.400
  Yes 0.222 0.333  0.444

     Jobsatisfaction
Y           No   Yes

  No  0.800 0.200
  Yes 0.333 0.667

     Desire
Y      Excellent  Fair

  No        0.600 0.400
  Yes       0.333 0.667
```

8 **<u>Predict the Outcome of “Enrolls” with the Testdata:</u>**

1. To use the predict function, type in the following:

```
> # predict with testdata
> results <- predict (model,testdata)
> # display results
> results
```

2. Review the results (Prediction for “Enrolls”) on the console. What is the prediction Yes or No?

```
> results
[1] Yes
Levels:  No Yes
```

9

<u>Review results</u>

1. Look at P(age=31-40 | Enrolls = No). You will observe a zero probability. Is this a problem?
   No, because the probability is not an absolute zero. The probability is just very small.

2. Build another NB model, with Laplace smoothing model2 = naiveBayes(Enrolls ~.,traindata, laplace=0.01)

```
> model1

Naive Bayes Classifier for Discrete Predictors

Call:
naiveBayes.default(x = X, y = Y, laplace = laplace)

A-priori probabilities:
Y
         No   Yes
0.000 0.357 0.643

Conditional probabilities:
     Age
Y         <=30 31 to 40      >40
       0.33333  0.33333 0.33333
  No   0.59841  0.00199 0.39960
  Yes  0.22259  0.44408 0.33333

     Income
Y      High   Low Medium
       0.333 0.333  0.333
  No   0.400 0.201  0.400
  Yes  0.223 0.333  0.444

     Jobsatisfaction
Y         No   Yes
       0.500 0.500
  No   0.799 0.201
  Yes  0.334 0.666

     Desire
Y      Excellent  Fair
           0.500 0.500
  No       0.600 0.400
  Yes      0.334 0.666

>
```

	3. Compare the probabilities here with those of the first model Note down your observations in the space provided below: The probabilities are very similar. The probability between 31 and 40 for the second model is slightly higher than the first model. In the second model there probability is measured at higher levels of precision with five numbers trailing the decimal point for age.

End of Lab Exercise