# Lab 10: Classification

NOTE: This lab has two parts:

* Part A: Naïve Bayesian Classifer
* Part B: Decision Trees

BOTH parts of the lab must be submitted for credit.

## **Part A: 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 |
|  | |
| **References:** | References used in this lab are located in your [***Student Resource Guide Appendix***](http://snhu-media.snhu.edu/files/course_repository/graduate/dat/dat510/dat_510_student_resource_guide_appendix.pdf)***.*** |

## **Part A – Building Naïve Bayesian Classifier**

### Workflow Overview

### LAB Instructions

|  |  |
| --- | --- |
| **Step** | **Action** |
| 1 | Download files needed for this lab from Blackboard:   * NBcoderev.R * Sample1.csv |
| 2 | **Set working directory and review training and test data**  1. Set the working directory using the following command:  **> setwd("<YOUR DIRECTORY>")**   * 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 | **Install and load library “e1071”**  Execute the following command to install the required packages and load the libraries:  **> install.packages("e1071", dependencies = TRUE)**  **> library("e1071")** |
| 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.   1. Execute the following command to display the data frames, to ensure they are loaded properly.   **> #Display data frames**  **> traindata**  **> testdata**   1. Review the output on the console window. |
| 5 | **Build the Naïve Bayesian classifier Model from First Principles:**   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**  Review the results of prior probabilities on the console   1. 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**   1. Do the same for the other variables.   **> incomeCounts <- table(traindata[,c("Enrolls", "Income")])**  **> incomeCounts <- incomeCounts/rowSums(incomeCounts)**  **> jsCounts <- table(traindata[,c("Enrolls", "Jobsatisfaction")])**  **> jsCounts<-jsCounts/rowSums(jsCounts)**  **> desireCounts <- table(traindata[,c("Enrolls", "Desire")])**  **> desireCounts <- desireCounts/rowSums(desireCounts)** |
| 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"]**   1. The prediction will be max(pyes,pno).   **> pyes**  **> pno**  **> max(pyes,pno)**   1. What is the predicted result for “Enrolls”?   The predicted result is yes. |
| 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**   1. Review the results on the console and compare these results to the **apriori probabilities** you manually computed earlier in step 5. |
| 8 | **Predict the Outcome of “Enrolls” with the Testdata:**   1. To use the predict function, type in the following:   **> # predict with testdata**  **> results <- predict (model,testdata)**  **> # display results**  **> results**   1. Review the results (Prediction for “Enrolls”) on the console. |
| 9 | **Review results**   1. Look at P(age=31-40 | Enrolls = No). You will observe a zero probability. Is this a problem?   The zero probability of P(age=31-40 | Enrolls = No) could be problem that might throw off the model accuracy. Since it’s a relatively small sample, there happened to be 0 observations of individuals aged 31-40 who did not enroll. However, P(age<=30 | Enrolls = No) equaled 60% and P(age>40 | Enrolls = No) equaled 40% so it is most likely that there would be individuals in the population between the ages of 31 and 40 who would not enroll, and the model would not be able to take this into account.   1. Build another NB model, with Laplace smoothing model2 = naiveBayes(Enrolls ~.,traindata, laplace=0.01) 2. Compare the probabilities here with those of the first model   Note down your observations in the space provided below:  With laplace smoothing incorporated in the model, there is now a small prior probability that someone between ages 31 and 40 would not enroll. The model shows P(age=31-40 | Enrolls = No) to equal 0.2%. This is only a small difference but I believe that it is much better than the zero probability that was initially observed. The overall probabilities remained constant, which is due to P(age<=30 | Enrolls = No) decreasing by about 0.16% and P(age>40 | Enrolls = No) decreasing by about 0.04%. |

*End of Lab Exercise*

## **Part B – Decision Trees**

|  |  |
| --- | --- |
| **Purpose:** | This lab is designed to investigate and practice Decision Tree (DT) models covered in the course work. After completing the tasks in this lab you should able to:   * Use R functions for Decision Tree models * Predict the outcome of an attribute based on the model |
|  | |
| **Tasks:** | Tasks you will complete in this lab exercise include:   * Use the R –Studio environment to code Decision Tree Models * Build a Decision Tree Model based on data whose schema is composed of attributes * Predict the outcome of one attribute based on the model |
|  | |
| **References:** | References used in this lab are located in your ***Student Resource Guide Appendix****.* |

### Workflow Overview

### LAB Instructions

|  |  |
| --- | --- |
| **Step** | **Action** |
| 1 | Download files needed for this lab from Blackboard:   * DT.R * DTdata.csv |
| 2 | **Set the Working Directory:**   1. Execute the command:   **setwd(“<YOUR DIRECTORY>")**  (Or use the “Tools” option in the tool bar in the RStudio environment.)   1. Load the package rpart.plot and the associated libraries. If prompted for the location to download select any integer representing a location nearest to you.   **> install.packages("rpart.plot", dependencies = TRUE)**  **> library("rpart")**  **> library("rpart.plot")** |
| 3 | **Read in the Data:**   * Use a data table with columns for data attributes : Play, Outlook, Temperature, Humidity and Windy * A Decision Tree allows for predicting the values of the attribute Play, given that you know the values for attributes like Outlook, Humidity and Windy.   1. Read in the data from the “Dtdata.csv” file in the working directory and display the contents:  **> #Read the data**  **> play\_decision <- read.table("DTdata.csv",header=TRUE,sep=",")**  **> play\_decision**   1. How many observations did you read in?   10 observations   1. How many variables (attributes) did you read in?   5 variables   1. Use the command “summary” for a detailed list of the table object you read in   **summary(play\_decision)**   1. Review the results. (The Summary is located in the console window.) |
| 4 | **Build the Decision Tree:**  Use the “rpart” package in R for classification by Decision Trees. The RPart Programs build classification or regression models of a very general structure using a two stage procedure; the resulting models can be represented as binary trees.   1. Use the following rpart commands to grow a Decision Tree:   **rpart (formula, data=, method=, control=)**   |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | |  |  | | --- | --- | | * **formula** | is in the format: outcome ~ predictor1+predictor2+predictor3+ect. | | * **data=** | specifies the dataframe | | * **method=** | "class" for a classification tree "anova" for a regression tree | | * **control=** | optional parameters for controlling tree growth. For example, control=rpart.control(minsplit=30, cp=0.001) requires that the minimum number of observations in a node be 30 before attempting a split and that a split must decrease the overall lack of fit by a factor of 0.001 (cost complexity factor) before being attempted. | | * **parms=** | Optional parameters for the splitting function. Anova splitting has no parameters. Poisson splitting has a single parameter, the coefficient of variation of the prior distribution on the rates. The default value is 1. Exponential splitting has the same parameter as Poisson.  For classification splitting, the list can contain any of: the vector of prior probabilities (component prior), the loss matrix (component loss) or the splitting index (component split). The priors must be positive and sum to 1. The loss matrix must have zeros on the diagonal and positive off-diagonal elements. The splitting index can be gini or information. The default priors are proportional to the data counts, the losses default to 1, and the split defaults to gini. | |   The "Play" attribute is the outcome that will be predicted.  2. Use the command:  **> fit <- rpart(Play ~ Outlook + Temperature + Humidity + Wind, method="class", data=play\_decision,**  **+ control=rpart.control(minsplit=1)**  **+ parms=list(split=’information’)**   1. You can now display “fit” and review the results:   **> summary(fit)**  Note that the leaf nodes information includes both the class label and the class probabilities  (P(no), P(yes)) |
| 5 | **Plot the Decision Tree:**   1. Review the arguments for rpart.plot function. Type in:   **> ?rpart.plot**  We will use the arguments “type” and “extra” in our plot.   1. Type in the following :   **> rpart.plot(fit, type=4, extra=1)**   1. Review the Decision Tree plot on the graphics window. |
| 6 | **Prepare Data to Test the Fitted Model:**  You must use “fit” for a new set of data to create predictions from the DT:  Play Decision Outlook Temperature Humidity Wind  ? rainy mild high FALSE   1. “newdata” is a data frame object and can be built for our test data. Type in the following statement:   **newdata <- data.frame(Outlook="rainy",Temperature="mild",Humidity="high",Wind=FALSE)**   1. Review the “newdata” displaying the dataframe   **> newdata**   1. The data displayed as follows:   **Outlook Temperature Humidity Wind**  **1 rainy mild high FALSE** |

|  |  |
| --- | --- |
| 7 | **Predict a Decision from the Fitted Model:**  The “predict” function is used to generate predictions from a fitted rpart object.   * “type” is a character string denoting the type of the predicted value * Use both “prob” and “class” to predict from a Decision Tree model   **predict(object, newdata = list(),**  **type = c("vector", "prob", "class", "matrix"))**  1. The **type=”prob”** gives the class probabilities for the prediction for newdata Type in  **> predict(fit,newdata=newdata,type="prob”)**   1. Repeat the prediction with type=”class”   **> predict(fit,newdata=newdata,type="class”)**  Review the results.   1. What is the prediction for the “newdata”?   The prediction for newdata is no. |

*End of Lab Exercise*