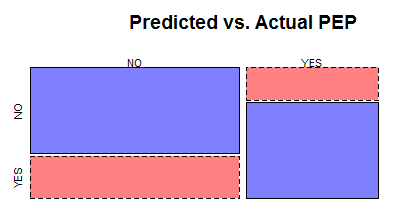
|  |
| --- |
| UMUC |
| Naïve Bayes Classification Using R |
| Optional Exercise |
|  |
| **DBST 667** |
|  |

|  |
| --- |
| In this exercise, you will run the Naïve Bayes classification method to predict if a customer would purchase a personal equity plan (PEP). The purchase predictors include age, gender, region, income and marital status, number of children, car ownership. You will also evaluate the classification accuracy. |



Contents

[Introduction 2](#_Toc448997422)

[Bank Dataset 2](#_Toc448997423)

[Launch the Program 3](#_Toc448997424)

[Load the Data 4](#_Toc448997425)

[Data Preprocessing 5](#_Toc448997426)

[Remove ID Variable 5](#_Toc448997427)

[Discretization 5](#_Toc448997428)

[Factor Function 6](#_Toc448997429)

[Preview the Variables Statistics 6](#_Toc448997430)

[Run Naïve Bayes Method 7](#_Toc448997431)

[Divide Data into Training and Test Set 7](#_Toc448997432)

[Use the Training Set to Build the Model 7](#_Toc448997433)

[Classification Accuracy for the Training Data 9](#_Toc448997434)

[Classification Accuracy for the Test Data 9](#_Toc448997435)

# Introduction

The purpose of this exercise is to build the Naïve Bayes classification model to determine the probability that the customer would purchase the Personal Equity Plan (PEP) based on age category, gender, region, income and marital status, number of children, car ownership, and existing accounts. PEP is a dependent variable with the class values YES and NO.

Naïve Bayes algorithm is a supervised classification method that assumes that all independent attributes in the dataset are conditionally independent. For example, while income and region equally influence the purchase decision, the income does not tell us anything about the region. Similarly, the region does not tell us anything about the income.

Naïve Bayes algorithm further assumes that all independent variables equally determine the value of dependent variable.

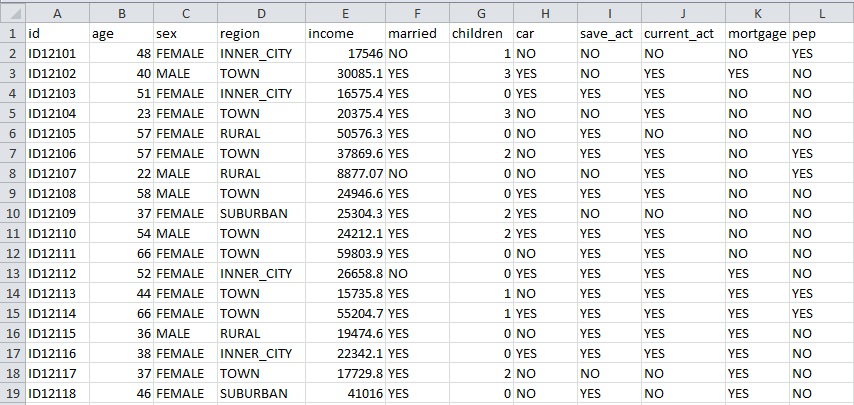
The algorithm output discussed below includes the number of instances in each class that have each attribute value. We use that data to compute the conditional probabilities for the attribute values of new instance. The output also includes the percentage of instances in each class, which is the probability of a class based on prior observation. The class of a new instance is the class with the highest calculated probability.

Depending on your operating system and an R version, your results might be slightly different.

# Bank Dataset

The bank dataset tracks the customer demographics, income, car ownership, accounts ownership, and personal equity loan purchase decision (PEP). Each customer has a unique identification number.

Figure 1 shows the partial content of the bank data file. The column headings in the first row of the file are the banking attribute names called variables. The remaining 600 rows are the data, where each row is a single banking record.



Variable names

Figure 1: Bank Data

# Launch the Program

Launch the R Studio program on Figure 2.

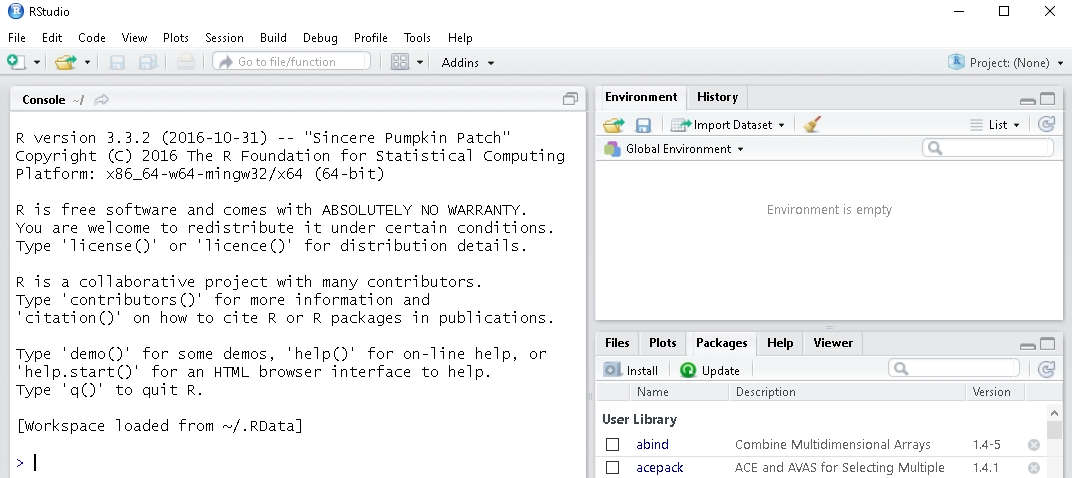


Figure 2: R Studio Interface

To run the Naïve Bayes method, you need to install the e1071 package. If you already installed this package, you may skip this step. If you have not installed the package, enter the following command into an application console and hit enter.

install.packages("e1071")

The command output on Figure 3 will display in the console window. An output specifies the package name and the hard drive location where the package was installed.

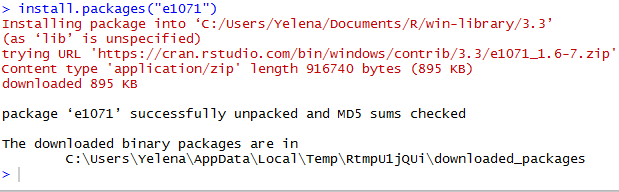


Figure 3: e1071 Package Installation Confirmation

You also need to install the arules package for running the discretization filter on numeric variables discussed below. If you already installed the arules package, you may skip this step. Otherwise, run the following command to install the package.

install.packages("arules")

Load the e1071 and arules packages into memory. This step needs to be repeated each time you restart R Studio.

Select the Packages tab in the bottom left window, and check the checkbox next to e1071 on Figure 4 and next to arules on Figure 5.

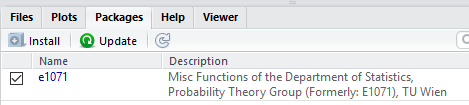


Figure 4: Load e1071 into Memory

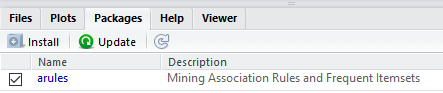


Figure 5: Load arules into memory

# Load the Data

Run the following read.csv command to load the data from bank.csv file into a Bank data frame.

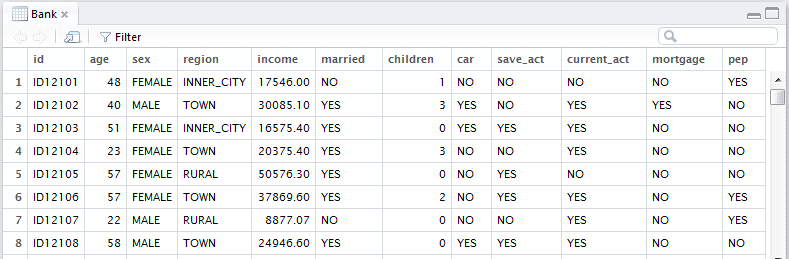
Bank <- read.csv("E:/Datasets/Bank.csv")

Run the following command to preview the Bank data.

View(Bank)

Each column in the data preview window on Figure **6** corresponds to a variable. The heading of each column is a variable name. The variable names match the column headings in the Bank.csv file.

Variables Income, age, and children are numeric. The first variable is a unique identifier, which we will remove during the data pre-processing.



Variables names

Data frame name

Figure 6: Bank Data Preview

# Data Preprocessing

## Remove ID Variable

The unique identifier values are irrelevant to the analysis. For example, we are not going to analyze the relationship between the id and the purchase decision. To remove the id variable, we set it equal to NULL

Bank$id<-NULL

## Discretization

The Naïve Bayes method requires all variable in the dataset to be discrete, or factor. However, variables age, income, and children are numeric.

To convert the age and income to factor variables, we run the unsupervised discretization filter with equal frequency binning. Numeric variable children has only 4 possible values in the dataset (0, 1, 2, 3). We use a factor function discussed in the next section.

Bank$age<-discretize(Bank$age, "frequency", categories=6)

summary(Bank$age)

Figure 7 shows the Age variable statistics after running the discretization filter, including the age ranges and the number of customers that fall into each age range.

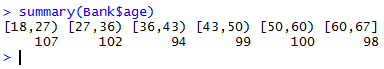


Figure 7: Age Variable Statistics after Running Discretization Filter

Bank$income<-discretize(Bank$income, "frequency", categories=6)

summary(Bank$income)

Figure 8 shows 6 income ranges and the number of customers with an income in each range.



Figure 8: Income Variable Statistics after Running Discretization Filter

## Factor Function

Run the factor function to convert the children attribute to factor.

Bank$children<-factor(Bank$children)

Run the following view command to preview the data after applying the data pre-processing filters. The data preview on **Figure 9** shows that the age values are the age ranges, and the income values are the income ranges.

View(Bank)

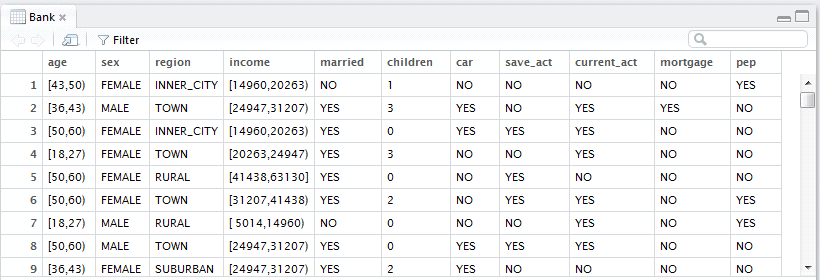


Figure 9: Data Preview after Preprocessing

## Preview the Variables Statistics

Run the summary command to preview the descriptive statistics for all variables after applying the data pre-processing filters.

summary(Bank)

Figure 10 shows the statistics for age and income variables are the value ranges and the number of instances with the value in each range. The statistics for children variable are number of instances with no children, number of instances with one child, etc.

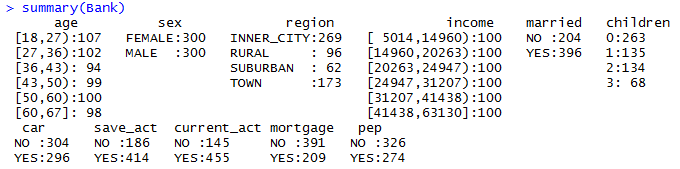


Figure 10: Bank Variables Statistics after Preprocessing

# Run Naïve Bayes Method

## Divide Data into Training and Test Set

We divide the data into training set and test set. We use the training set to build the classification tree model, and we use the test set to evaluate the accuracy of a model. The training set will contain 70% of data, and the test set will contain 30% of data.

Setting the seed value enables reproducing the results when the method is rerun.

set.seed(1234)

ind <- sample(2, nrow(Bank), replace = TRUE, prob = c(0.7, 0.3))

train.data <- Bank[ind == 1, ]

test.data <- Bank[ind == 2, ]

## Use the Training Set to Build the Model

The naiveBayes function takes the formula and a dataset name as an input. A formula is an expression that contains the dependent followed by ~ (tilde) and independent variables. The dependent variables are delimited by + (plus sign)

For example, pep~children+sex means that pep is a dependent variable, and children and sex are the independent variables.

To use all remaining variables as predictors, we may use a wildcard character . (dot) instead of listing all variable names.

Run the following command to create a model using the training data where pep is the dependent variable and all remaining variables are independent variables. Store the output in the variable model.

model<-naiveBayes(pep~., train.data)

Run the following print command to display the output stored in the model variable.

print(model)

Figure 11 shows the partial output. The first sentence in the output means that all independent variables are factors. The A-priori probabilities section shows the probability that pep variable=YES and the probability that PED variable=NO.

The remaining output shows conditional probabilities for each independent variable. Each row label is a PEP value, and each column is a value of the independent variable. An intersection of a row and column is the probability of the corresponding independent variable value given the corresponding value of depended variable.

For example, the conditional probabilities for age variables show that the probability of an age range 18-27 is approximately 0.23 given PEP=NO . The probability of ab age range 18-27 is approximately 0.13 given PEP=YES.

The numbers in each row add to a 1.

To predict the probability that the PEP is YES, we multiply the corresponding conditional probabilities for the values of all independent variables and divide the product by the Apriori probability of PEP=YES. (0.4423007)

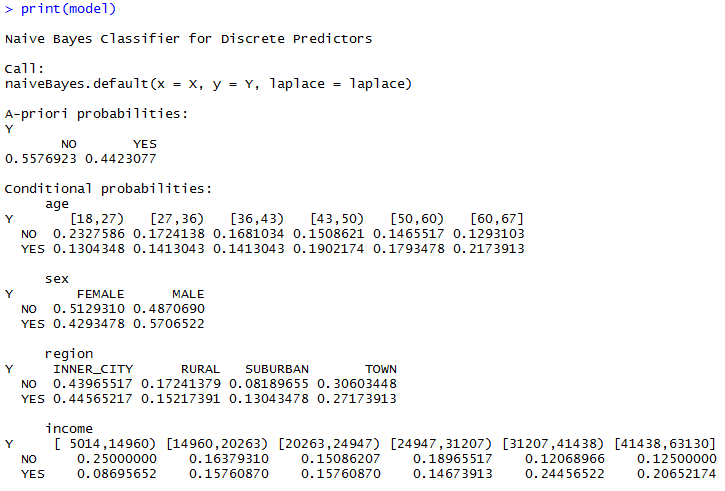


Figure 11: Partial Output

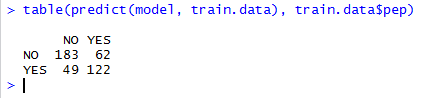
## Classification Accuracy for the Training Data

Run the following table command to build a confusion matrix for the training set on Figure 12. The **predict** command uses the model to predict the pep value for the instances. It takes the model and the dataset name as an input. For the dataset, we use train.data because it’s the training dataset we used to build the model.

The table command takes the predicted class (output of predict command) and an actual class as an input.

table(predict(model, train.data), train.data$pep)

A **Confusion matrix** shows how many bank records in the test data have been assigned to each class. For each matrix element, the row label is a predicted class, and the column label is an actual class. The number of correctly classified instances is the sum of numbers on diagonal from top left to bottom right. The sum of numbers outside the diagonal from top left to bottom right is the number of misclassified instances. The sum of all matrix entries is the number of instances in the training set.



The number of correctly classified instances= 183+122=305

The number of misclassified instances=62+49=111

The number of instances in the training set=305+111=416

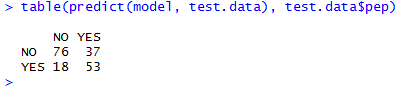
The classification accuracy is sum of numbers on diagonal/sum of all numbers=305/416 **73%**

Figure 12: Confusion matrix for training data

## Classification Accuracy for the Test Data

Run the following commands to evaluate the model for the test data and to build a confusion matrix on Figure 13. For the prediction to work, the values of the categorical variables (levels) in the test data must be the same as the values in the training data. In addition, the number of variables and variable names in the training set need to match the number of variables and the variable names in the test set.

table(predict(model, test.data), test.data$pep)



The number of correctly classified instances= 76+53=129

The number of misclassified instances=37+18=55

The number of instances in the test set=129+55=184

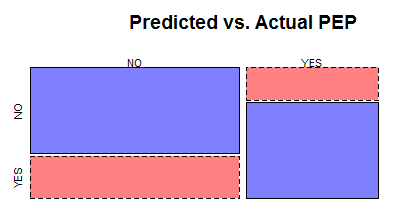
The classification accuracy is sum of numbers on diagonal/sum of all numbers=129/184 **70%**

Figure 13: Confusion matrix for test data

Run the following command to build the mosaic plot for predicted vs. actual PEP values on Figure 14. The mosaicplot function takes the confusion matrix table as the first parameter. The second parameter shade=TRUE means that we want to use different colors. Main parameter value is the plot title.

mosaicplot(table(predict(model, test.data), test.data$pep), shade=TRUE, main="Predicted vs. Actual PEP")

The blue color represents the proportion of instances with the predicted class=actual class. The red color represents the proportion of misclassified instances.



Predicted PEP=YES

Actual PEP=YES

Predicted PEP=NO

Actual PEP=Yes

Predicted PEP=YES

Actual PEP=NO

Predicted PEP=NO

Actual PEP=NO

Figure 14: Predicted VS Actual PEP plot