Practicum 2

Smit Patil

7/2/2020

---Problem 1---

1. Download the data set Census Income Data for Adults along with its explanation. Note that the data file does not contain header names; you may wish to add those. The description of each column can be found in the data set explanation.

```
#Importing adult data
adult_data <- read.csv("adult.data", header = F)

#Creating data frame to add column names to the adult data
adult.names <- c("age","workclass","fnlwgt","education","education_num","maritial_status","occupation",

#Adding names to the adult data
colnames(adult_data) <- adult.names
```

2. Explore the data set as you see fit and that allows you to get a sense of the data and get comfortable with it.

```
#Importing Libraries to perform data cleaning
library(tidyverse)
```

```
## -- Attaching packages ---------------------------------------------------------------------------

## v ggplot2 3.3.0     v purrr   0.3.4
## v tibble  3.0.1     v dplyr   0.8.5
## v tidyr   1.0.3     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0

## -- Conflicts ------------------------------------------------------------------------------------
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(arules)
```

```
## Warning: package 'arules' was built under R version 4.0.2

## Loading required package: Matrix
```

```
##
## Attaching package: 'Matrix'

## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack

##
## Attaching package: 'arules'

## The following object is masked from 'package:dplyr':
##
##     recode

## The following objects are masked from 'package:base':
##
##     abbreviate, write
```

```
#Creating a copy of the adult data
adult.data <- adult_data

#Looking at the head and the structure of the data
head(adult.data)
```

```
##   age         workclass fnlwgt  education education_num     maritial_status
## 1  39         State-gov  77516  Bachelors            13       Never-married
## 2  50  Self-emp-not-inc  83311  Bachelors            13  Married-civ-spouse
## 3  38           Private 215646    HS-grad             9            Divorced
## 4  53           Private 234721       11th             7  Married-civ-spouse
## 5  28           Private 338409  Bachelors            13  Married-civ-spouse
## 6  37           Private 284582    Masters            14  Married-civ-spouse
##           occupation   relationship   race     sex capital_gain capital_loss
## 1       Adm-clerical  Not-in-family  White    Male         2174            0
## 2    Exec-managerial        Husband  White    Male            0            0
## 3  Handlers-cleaners  Not-in-family  White    Male            0            0
## 4  Handlers-cleaners        Husband  Black    Male            0            0
## 5     Prof-specialty           Wife  Black  Female            0            0
## 6    Exec-managerial           Wife  White  Female            0            0
##   hours_per_week native_country salary
## 1             40  United-States  <=50K
## 2             13  United-States  <=50K
## 3             40  United-States  <=50K
## 4             40  United-States  <=50K
## 5             40           Cuba  <=50K
## 6             40  United-States  <=50K
```

```
str(adult.data)
```

```
## 'data.frame':    32561 obs. of  15 variables:
##  $ age            : int  39 50 38 53 28 37 49 52 31 42 ...
##  $ workclass      : chr  " State-gov" " Self-emp-not-inc" " Private" " Private" ...
##  $ fnlwgt         : int  77516 83311 215646 234721 338409 284582 160187 209642 45781 159449 ...
```

```
##  $ education      : chr  " Bachelors" " Bachelors" " HS-grad" " 11th" ...
##  $ education_num  : int  13 13 9 7 13 14 5 9 14 13 ...
##  $ maritial_status: chr  " Never-married" " Married-civ-spouse" " Divorced" " Married-civ-spouse" ..
##  $ occupation     : chr  " Adm-clerical" " Exec-managerial" " Handlers-cleaners" " Handlers-cleaners
##  $ relationship   : chr  " Not-in-family" " Husband" " Not-in-family" " Husband" ...
##  $ race           : chr  " White" " White" " White" " Black" ...
##  $ sex            : chr  " Male" " Male" " Male" " Male" ...
##  $ capital_gain   : int  2174 0 0 0 0 0 0 0 14084 5178 ...
##  $ capital_loss   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ hours_per_week : int  40 13 40 40 40 40 16 45 50 40 ...
##  $ native_country : chr  " United-States" " United-States" " United-States" " United-States" ...
##  $ salary         : chr  " <=50K" " <=50K" " <=50K" " <=50K" ...
```

```
#Summarising the adult data
summary(adult.data)
```

```
##       age         workclass              fnlwgt          education
##  Min.   :17.00   Length:32561       Min.   :  12285   Length:32561
##  1st Qu.:28.00   Class :character   1st Qu.: 117827   Class :character
##  Median :37.00   Mode  :character   Median : 178356   Mode  :character
##  Mean   :38.58                      Mean   : 189778
##  3rd Qu.:48.00                      3rd Qu.: 237051
##  Max.   :90.00                      Max.   :1484705
##  education_num   maritial_status     occupation         relationship
##  Min.   : 1.00   Length:32561       Length:32561       Length:32561
##  1st Qu.: 9.00   Class :character   Class :character   Class :character
##  Median :10.00   Mode  :character   Mode  :character   Mode  :character
##  Mean   :10.08
##  3rd Qu.:12.00
##  Max.   :16.00
##      race               sex             capital_gain    capital_loss
##  Length:32561       Length:32561       Min.   :    0   Min.   :   0.0
##  Class :character   Class :character   1st Qu.:    0   1st Qu.:   0.0
##  Mode  :character   Mode  :character   Median :    0   Median :   0.0
##                                        Mean   : 1078   Mean   :  87.3
##                                        3rd Qu.:    0   3rd Qu.:   0.0
##                                        Max.   :99999   Max.   :4356.0
##  hours_per_week  native_country        salary
##  Min.   : 1.00   Length:32561       Length:32561
##  1st Qu.:40.00   Class :character   Class :character
##  Median :40.00   Mode  :character   Mode  :character
##  Mean   :40.44
##  3rd Qu.:45.00
##  Max.   :99.00
```

```
#We see that there are unwanted characters in the data, so we replace thode with NA
adult.data[adult.data == " ?"] <- NA

#Finding the column names which have  NA values present in them
colnames(adult.data)[colSums(is.na(adult.data)) > 0]
```

```
## [1] "workclass"      "occupation"     "native_country"
```

```r
#Replacing the NA values in the columns with the most common value
common.workclass <- names(which.max(table(adult.data$workclass)))
adult.data$workclass <- adult.data$workclass %>% replace_na(common.workclass)

common.occupation <- names(which.max(table(adult.data$occupation)))
adult.data$occupation <- adult.data$occupation %>% replace_na(common.occupation)

common.native_country <- names(which.max(table(adult.data$native_country)))
adult.data$native_country <- adult.data$native_country %>% replace_na(common.native_country)

#Verifying wheather all the NA values are replaced
anyNA(adult.data)
```

```
## [1] FALSE
```

```r
#Dicretizing the age column in the adult data into 4 discrete bins, so that each bin reprensents 25% of
adult.data$age <- discretize(adult.data$age, breaks = 4)

#Creating factors for sex and salary group in the adult dataset
adult.data$sex <- as.factor(adult.data$sex)
adult.data$salary <- as.factor(adult.data$salary)
```

3. Split the data set 75/25 so you retain 25% for testing using random sampling.

```r
#Importing libraries to randomly split the dataset
library(caret)
```

```
## Loading required package: lattice

##
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':
##
##     lift
```

```r
set.seed(1)

#Spliting the dataset using createDataPartition() function, so that the data is samped in equal distrib
adult.sample <- createDataPartition(adult.data$salary, p = 0.25, list = FALSE, times = 1)

#Assigining 25% of the data for testing and the remaining 75% for training
adult.testing <- adult.data[adult.sample,]
adult.training <- adult.data[-adult.sample,]
```

4. Using the Naive Bayes Classification algorithm from the KlaR, naivebayes, and e1071 packages, build an ensemble classifier that predicts whether an individual earns more than or less than US$50,000. Only use the features age, education, workclass, sex, race, and native-country. Ignore any other features in your model. You need to transform continuous variables into categorical variables by binning (use equal size bins from in to max). Note that some packages might not work with your current version of R and may need to be downgraded.

```r
#Importing Libraries to perform Naive Bayes using KlaR, naivebayes, and e1071 packages
library(MASS)
```

```
##
## Attaching package: 'MASS'

## The following object is masked from 'package:dplyr':
##
##     select
```

```r
library(klaR)
library(naivebayes)
```

```
## Warning: package 'naivebayes' was built under R version 4.0.2

## naivebayes 0.9.7 loaded
```

```r
library(e1071)
library(gmodels)

#Selecting limited features for training and testing Naive Bayes
features.train <- adult.training[c("age","education","workclass","sex","race","native_country","salary"
features.test <- adult.testing[c("age","education","workclass","sex","race","native_country","salary")]

#Training Naive Bayes model using the klaR package
nb.klaR <- NaiveBayes(salary~., data = features.train)

#Training Naive Bayes model using the naivebayes package
nb.naivebayes <- naive_bayes(salary~., features.train)
```

```
## Warning: naive_bayes(): Feature education - zero probabilities are present.
## Consider Laplace smoothing.

## Warning: naive_bayes(): Feature workclass - zero probabilities are present.
## Consider Laplace smoothing.

## Warning: naive_bayes(): Feature native_country - zero probabilities are present.
## Consider Laplace smoothing.
```

```r
#Training Naive Bayes model using the e1071 package
nb.e1071 <- naiveBayes(salary~., data = features.train)

#Creating an ensemble model of all the three packages i.e. KlaR, naivebayes, and e1071
ensemble.model_1 <- function(data)
  {
    klaR.prediction <- predict(nb.klaR, data)[[1]]
    naivebayes.prediction <- predict(nb.naivebayes, data)
    e1071.prediction <- predict(nb.e1071, data)
    nb.ensemble_1 <- data.frame("klaR" = klaR.prediction, "naivebayes" = naivebayes.prediction, "e1071"
                                "Majority_Vote" =
```

```r
                                  as.factor(ifelse(klaR.prediction == ' >50K' & naivebayes.prediction ==
                                            ifelse(klaR.prediction == ' >50K' & e1071.prediction == ' >5
                                            ifelse(naivebayes.prediction == ' >50K' & e1071.prediction =
                                                   ' <=50K')))))
    return(nb.ensemble_1)
}

#Predicting the salary group of the test data with help of the ensemble model
ensemble.prediction_1 <- ensemble.model_1(features.test[,-7])
```

```
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 8

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 22

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 248

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 287

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 291

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 370

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 664

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 699

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 766

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 915

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 1148

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 1190

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 1236
```

```
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 1258

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 1326

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 1372

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 1409

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 1424

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 1475

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 1515

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 1562

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 1704

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 1757

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 1764

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 1796

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 1831

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 1842

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 1853

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 2050

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 2061
```

```
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 2065

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 2275

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 2319

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 2412

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 2413

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 2451

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 2460

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 2707

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 2793

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 2843

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 2942

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 3067

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 3071

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 3114

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 3420

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 3441

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 3484
```

```
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 3528

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 3578

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 3592

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 3622

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 3655

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 3694

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 3876

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 3918

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 4046

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 4076

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 4151

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 4153

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 4196

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 4262

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 4300

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 4340

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 4401
```

```
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 4456

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 4457

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 4514

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 4559

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 4583

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 4638

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 4640

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 4716

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 4724

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 4744

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 4789

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 4832

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 4886

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 4976

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 5068

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 5109

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 5261
```

```
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 5413

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 5557

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 5594

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 5819

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 5886

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 6011

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 6031

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 6041

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 6067

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 6116

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 6195

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 6260

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 6285

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 6350

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 6408

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 6506

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 6526
```

```
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 6714

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 6892

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 7120

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 7145

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 7263

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 7270

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 7305

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 7409

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 7437

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 7557

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 7722

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 7736

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 7764

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 7819

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 7881

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 7951

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 7986
```

```
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 8070

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 8087

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 8118

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 8136
```

```r
#Printing the CrossTable to display the predictions from the actual value
CrossTable(ensemble.prediction_1$Majority_Vote, features.test$salary, dnn = c('predicted','actual'))
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## | Chi-square contribution |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  8141
##
##
##              | actual
##    predicted |     <=50K |      >50K | Row Total |
## -------------|-----------|-----------|-----------|
##        <=50K |      5690 |      1108 |      6798 |
##              |    54.330 |   171.218 |           |
##              |     0.837 |     0.163 |     0.835 |
##              |     0.921 |     0.565 |           |
##              |     0.699 |     0.136 |           |
## -------------|-----------|-----------|-----------|
##         >50K |       490 |       853 |      1343 |
##              |   275.007 |   866.671 |           |
##              |     0.365 |     0.635 |     0.165 |
##              |     0.079 |     0.435 |           |
##              |     0.060 |     0.105 |           |
## -------------|-----------|-----------|-----------|
## Column Total |      6180 |      1961 |      8141 |
##              |     0.759 |     0.241 |           |
## -------------|-----------|-----------|-----------|
##
##
```

```r
#Calculating the accuracy of the ensemble model with the help the CrossTable
accuracy.ensemble_1 <- ((5623+885)/(8140))*100
sprintf("The accuracy of the Naive Bayes model using ensemble model is %s percent", accuracy.ensemble_1)
```

```
## [1] "The accuracy of the Naive Bayes model using ensemble model is 79.95085995086 percent"
```

5. Create a full logistic regression model of the same features as in (4) (i.e., do not eliminate any features regardless of p-value). Be sure to either use dummy coding for categorical features or convert them to factor variables and ensure that the glm function does the dummy coding. Add the logistic regression model to the ensemble built in (4).

```r
#Creatin a Logistic Regression model for the same training data
glm.lr <- glm(salary~., data = features.train, family = binomial)

#Predicting the Salary group based on the above Logistic Regression model
lr.prediction <- ifelse(predict(glm.lr, newdata = features.test, type = "response") < 0.5, " <=50K"," >5
```

6. Add the logistic regression model to the ensemble built in (4).

```r
#Adding Logistic Regression to the above ensemble model
ensemble.model_2 <- function(data)
  {
    klaR.prediction <- predict(nb.klaR, data)[[1]]
    naivebayes.prediction <- predict(nb.naivebayes, data)
    e1071.prediction <- predict(nb.e1071, data)
    lr.prediction <- ifelse(predict(glm.lr, newdata = data, type = "response") < 0.5, " <=50K"," >50K")
    nb.ensemble_2 <- data.frame("klaR" = klaR.prediction, "naivebayes" = naivebayes.prediction, "e1071"
                                "Logistic_Regression" = lr.prediction,
                                "Majority_Vote" =
                                  as.factor(ifelse(klaR.prediction == ' >50K' & naivebayes.prediction =
                                                   e1071.prediction == ' >50K', ' >50K',
                                         ifelse(klaR.prediction == ' >50K' & naivebayes.prediction =
                                                   lr.prediction == ' >50K', ' >50K',
                                         ifelse(naivebayes.prediction == ' >50K' & e1071.prediction =
                                                   lr.prediction == ' >50K', ' >50K', ' <=50K')))))
    return(nb.ensemble_2)
  }

#Predicting the salary group of the test data with help of the new ensemble model
ensemble.prediction_2 <- ensemble.model_2(features.test[,-7])
```

```
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 8

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 22

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 248

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 287
```

```
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 291

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 370

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 664

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 699

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 766

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 915

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 1148

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 1190

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 1236

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 1258

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 1326

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 1372

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 1409

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 1424

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 1475

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 1515

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 1562
```

```
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 1704

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 1757

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 1764

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 1796

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 1831

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 1842

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 1853

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 2050

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 2061

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 2065

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 2275

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 2319

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 2412

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 2413

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 2451

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 2460

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 2707
```

```
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 2793

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 2843

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 2942

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 3067

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 3071

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 3114

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 3420

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 3441

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 3484

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 3528

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 3578

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 3592

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 3622

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 3655

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 3694

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 3876

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 3918
```

```
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 4046

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 4076

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 4151

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 4153

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 4196

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 4262

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 4300

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 4340

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 4401

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 4456

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 4457

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 4514

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 4559

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 4583

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 4638

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 4640

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 4716
```

```
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 4724

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 4744

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 4789

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 4832

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 4886

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 4976

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 5068

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 5109

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 5261

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 5413

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 5557

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 5594

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 5819

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 5886

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 6011

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 6031

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 6041
```

```
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 6067

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 6116

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 6195

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 6260

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 6285

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 6350

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 6408

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 6506

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 6526

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 6714

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 6892

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 7120

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 7145

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 7263

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 7270

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 7305

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 7409
```

```
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 7437

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 7557

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 7722

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 7736

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 7764

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 7819

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 7881

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 7951

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 7986

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 8070

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 8087

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 8118

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 8136
```

```r
#Printing the CrossTable to display the predictions from the actual value of the new ensemble model
CrossTable(ensemble.prediction_2$Majority_Vote, features.test$salary, dnn = c('predicted','actual'))
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## | Chi-square contribution |
## |           N / Row Total |
## |           N / Col Total |
```

```
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  8141
##
##
##              | actual
##    predicted |      <=50K |       >50K |  Row Total |
## -------------|-----------|-----------|-----------|
##        <=50K |       5832 |       1193 |       7025 |
##              |     46.726 |    147.253 |            |
##              |      0.830 |      0.170 |      0.863 |
##              |      0.944 |      0.608 |            |
##              |      0.716 |      0.147 |            |
## -------------|-----------|-----------|-----------|
##         >50K |        348 |        768 |       1116 |
##              |    294.128 |    926.932 |            |
##              |      0.312 |      0.688 |      0.137 |
##              |      0.056 |      0.392 |            |
##              |      0.043 |      0.094 |            |
## -------------|-----------|-----------|-----------|
## Column Total |       6180 |       1961 |       8141 |
##              |      0.759 |      0.241 |            |
## -------------|-----------|-----------|-----------|
##
##
```

```r
#Calculating the accuracy of the new ensemble model with the help the CrossTable
accuracy.ensemble_1 <- ((5741+831)/(8140))*100
sprintf("The accuracy of the Naive Bayes model using ensemble model is %s percent", accuracy.ensemble_1)
```

```
## [1] "The accuracy of the Naive Bayes model using ensemble model is 80.7371007371007 percent"
```

7. Using the ensemble model from (6), predict whether a 35-year-old white female adult who is a local government worker with a doctorate who immigrated from Portugal earns more or less than US$50,000.

```r
#Creating a new data frame for the new adult for predection
new_adult <- data.frame(35, " Doctorate", " Local-gov", " Female", " White", " Portugal", NA)

#Creating a data frame and adding the column names to the new adult data
data.col_names <- c("age","education","workclass","sex","race","native_country", "salary")
colnames(new_adult) <- data.col_names

#Creating another sample of the adult data to perform factor operation on the new adult data
new_adult.data <- adult_data[c("age","education","workclass","sex","race","native_country", "salary")]

#Binding the row of the new adult data with the original adult data
new_adult.data <- rbind(new_adult, new_adult.data)

#Dicretizing the age column in the adult data into 4 discrete bins, so that each bin reprensents 25% of
new_adult.data$age <- discretize(new_adult.data$age, breaks = 4)
```

```r
#Creating factors for sex and salary group in the adult dataset
new_adult.data$sex <- as.factor(new_adult.data$sex)
new_adult.data$salary <- as.factor(new_adult.data$salary)

#Retiving the new adult data after performing the factor opearions
new_testing.data <- new_adult.data[1,]

#Predicting the salary range for the new adult using the ensemle model
ensemble.prediction_3 <- ensemble.model_2(new_testing.data[,-7])
```

```
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 1
```

```r
sprintf("The predicted salary range for the given adult data using ensemble model is %s", ensemble.predi
```

```
## [1] "The predicted salary range for the given adult data using ensemble model is  <=50K"
```

8. Calculate accuracy and prepare confusion matrices for all three Bayes implementations (KlaR, naivebayes, e1071) and the logistic regression model. Compare the implementations and comment on differences. Be sure to use the same training data set for all three. The results should be the same but they may differ if the different implementations deal differently with LaPalace Estimators.

```r
#Creating prediction lables to implement confusionMatrix
prediction_labels <- features.test$salary

#klar Package
#Predicting the Salary range using klar package
klaR.prediction <- predict(nb.klaR, features.test[,-7])
```

```
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 8

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 22

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 248

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 287

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 291

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 370

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 664
```

```
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 699

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 766

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 915

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 1148

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 1190

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 1236

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 1258

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 1326

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 1372

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 1409

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 1424

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 1475

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 1515

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 1562

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 1704

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 1757

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 1764
```

```
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 1796

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 1831

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 1842

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 1853

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 2050

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 2061

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 2065

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 2275

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 2319

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 2412

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 2413

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 2451

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 2460

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 2707

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 2793

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 2843

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 2942
```

```
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 3067

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 3071

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 3114

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 3420

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 3441

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 3484

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 3528

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 3578

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 3592

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 3622

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 3655

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 3694

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 3876

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 3918

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 4046

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 4076

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 4151
```

```
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 4153

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 4196

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 4262

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 4300

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 4340

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 4401

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 4456

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 4457

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 4514

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 4559

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 4583

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 4638

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 4640

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 4716

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 4724

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 4744

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 4789
```

```
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 4832

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 4886

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 4976

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 5068

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 5109

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 5261

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 5413

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 5557

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 5594

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 5819

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 5886

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 6011

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 6031

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 6041

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 6067

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 6116

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 6195
```

```
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 6260

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 6285

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 6350

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 6408

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 6506

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 6526

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 6714

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 6892

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 7120

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 7145

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 7263

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 7270

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 7305

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 7409

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 7437

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 7557

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 7722
```

```
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 7736

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 7764

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 7819

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 7881

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 7951

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 7986

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 8070

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 8087

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 8118

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 8136
```

```r
#Printing the CrossTable to display the predictions from the actual value of the klaR package
CrossTable(klaR.prediction$class, features.test$salary, dnn = c('predicted','actual'))
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## | Chi-square contribution |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  8141
##
##
##              | actual
##    predicted |     <=50K |      >50K | Row Total |
## -------------|-----------|-----------|-----------|
```

```
##        <=50K |      5992 |      1503 |      7495 |
##              |    16.072 |    50.649 |           |
##              |     0.799 |     0.201 |     0.921 |
##              |     0.970 |     0.766 |           |
##              |     0.736 |     0.185 |           |
## -------------|-----------|-----------|-----------|
##         >50K |       188 |       458 |       646 |
##              |   186.465 |   587.635 |           |
##              |     0.291 |     0.709 |     0.079 |
##              |     0.030 |     0.234 |           |
##              |     0.023 |     0.056 |           |
## -------------|-----------|-----------|-----------|
## Column Total |      6180 |      1961 |      8141 |
##              |     0.759 |     0.241 |           |
## -------------|-----------|-----------|-----------|
##
##
```

```
#Calculating the accuracy of the klaR package with the help the CrossTable
accuracy.klaR <- ((5805+702)/(8140))*100
sprintf("The accuracy of the Naive Bayes model using klaR package is %s percent",accuracy.klaR)
```

```
## [1] "The accuracy of the Naive Bayes model using klaR package is 79.9385749385749 percent"
```

```
#Generating the confusionMatrix for the klaR package
confusionMatrix(klaR.prediction$class, prediction_labels)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  <=50K  >50K
##      <=50K   5992  1503
##      >50K     188   458
##
##                Accuracy : 0.7923
##                  95% CI : (0.7833, 0.8011)
##     No Information Rate : 0.7591
##     P-Value [Acc > NIR] : 6.228e-13
##
##                   Kappa : 0.2634
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9696
##             Specificity : 0.2336
##          Pos Pred Value : 0.7995
##          Neg Pred Value : 0.7090
##              Prevalence : 0.7591
##          Detection Rate : 0.7360
##    Detection Prevalence : 0.9206
##       Balanced Accuracy : 0.6016
##
##        'Positive' Class :  <=50K
##
```

```r
#naivebayes Package
#Predicting the Salary range using naivebayes package
naivebayes.prediction <- predict(nb.naivebayes, features.test[,-7])

#Printing the CrossTable to display the predictions from the actual value of the naivebayes package
CrossTable(naivebayes.prediction, features.test$salary, dnn = c('predicted','actual'))
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## | Chi-square contribution |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  8141
##
##
##              | actual
##    predicted |     <=50K |      >50K | Row Total |
## -------------|-----------|-----------|-----------|
##        <=50K |      5690 |      1108 |      6798 |
##              |    54.330 |   171.218 |           |
##              |     0.837 |     0.163 |     0.835 |
##              |     0.921 |     0.565 |           |
##              |     0.699 |     0.136 |           |
## -------------|-----------|-----------|-----------|
##         >50K |       490 |       853 |      1343 |
##              |   275.007 |   866.671 |           |
##              |     0.365 |     0.635 |     0.165 |
##              |     0.079 |     0.435 |           |
##              |     0.060 |     0.105 |           |
## -------------|-----------|-----------|-----------|
## Column Total |      6180 |      1961 |      8141 |
##              |     0.759 |     0.241 |           |
## -------------|-----------|-----------|-----------|
##
##
```

```r
#Calculating the accuracy of the naivebayes package with the help the CrossTable
accuracy.naivebayes <- ((5623+885)/(8140))*100
sprintf("The accuracy of the Naive Bayes model using naivebayes package is %s percent",accuracy.naiveba
```

```
## [1] "The accuracy of the Naive Bayes model using naivebayes package is 79.95085995086 percent"
```

```r
#Generating the confusionMatrix for the naivebayes package
confusionMatrix(naivebayes.prediction, prediction_labels)
```

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  <=50K  >50K
##      <=50K   5690  1108
##      >50K     490   853
##
##                Accuracy : 0.8037
##                  95% CI : (0.7949, 0.8123)
##     No Information Rate : 0.7591
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3986
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9207
##             Specificity : 0.4350
##          Pos Pred Value : 0.8370
##          Neg Pred Value : 0.6351
##              Prevalence : 0.7591
##          Detection Rate : 0.6989
##    Detection Prevalence : 0.8350
##       Balanced Accuracy : 0.6778
##
##        'Positive' Class :  <=50K
##
```

```
#e1071 Package
#Predicting the Salary range using e1071 package
e1071.prediction <- predict(nb.e1071, features.test[,-7])

#Printing the CrossTable to display the predictions from the actual value of the e1071 package
CrossTable(e1071.prediction, features.test$salary, dnn = c('predicted','actual'))
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## | Chi-square contribution |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  8141
##
##
##              | actual
##    predicted |     <=50K |      >50K | Row Total |
## -------------|-----------|-----------|-----------|
##        <=50K |      5690 |      1108 |      6798 |
```

```
##              |    54.330 |   171.218 |           |
##              |     0.837 |     0.163 |     0.835 |
##              |     0.921 |     0.565 |           |
##              |     0.699 |     0.136 |           |
## -------------|-----------|-----------|-----------|
##         >50K |       490 |       853 |      1343 |
##              |   275.007 |   866.671 |           |
##              |     0.365 |     0.635 |     0.165 |
##              |     0.079 |     0.435 |           |
##              |     0.060 |     0.105 |           |
## -------------|-----------|-----------|-----------|
## Column Total |      6180 |      1961 |      8141 |
##              |     0.759 |     0.241 |           |
## -------------|-----------|-----------|-----------|
##
##
```

```
#Calculating the accuracy of the e1071 package with the help the CrossTable
accuracy.e1071 <- ((6191+1947)/(8140))*100
sprintf("The accuracy of the Naive Bayes model using e1071 package is %s percent",accuracy.e1071)
```

```
## [1] "The accuracy of the Naive Bayes model using e1071 package is 99.97542997543 percent"
```

```
#Generating the confusionMatrix for the e1071 package
confusionMatrix(e1071.prediction, prediction_labels)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  <=50K  >50K
##      <=50K   5690  1108
##      >50K     490   853
##
##                Accuracy : 0.8037
##                  95% CI : (0.7949, 0.8123)
##     No Information Rate : 0.7591
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3986
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9207
##             Specificity : 0.4350
##          Pos Pred Value : 0.8370
##          Neg Pred Value : 0.6351
##              Prevalence : 0.7591
##          Detection Rate : 0.6989
##    Detection Prevalence : 0.8350
##       Balanced Accuracy : 0.6778
##
##        'Positive' Class :  <=50K
##
```

```r
#Logistic Regression
#Predicting the Salary range using Logistic Regression
lr.prediction <- ifelse(predict(glm.lr, newdata = features.test, type = "response") < 0.5, " <=50K"," >5

#Printing the CrossTable to display the predictions from the actual value of the logistic regression
CrossTable(lr.prediction, features.test$salary, dnn = c('predicted','actual'))
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## | Chi-square contribution |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  8141
##
##
##              | actual
##    predicted |     <=50K |      >50K | Row Total |
## -------------|-----------|-----------|-----------|
##        <=50K |      5840 |      1198 |      7038 |
##              |    46.291 |   145.883 |           |
##              |     0.830 |     0.170 |     0.865 |
##              |     0.945 |     0.611 |           |
##              |     0.717 |     0.147 |           |
## -------------|-----------|-----------|-----------|
##         >50K |       340 |       763 |      1103 |
##              |   295.371 |   930.848 |           |
##              |     0.308 |     0.692 |     0.135 |
##              |     0.055 |     0.389 |           |
##              |     0.042 |     0.094 |           |
## -------------|-----------|-----------|-----------|
## Column Total |      6180 |      1961 |      8141 |
##              |     0.759 |     0.241 |           |
## -------------|-----------|-----------|-----------|
##
##
```

```r
#Calculating the accuracy of the logistic regression with the help the CrossTable
accuracy.lr <- ((5776+806)/(8140))*100
sprintf("The accuracy of the Naive Bayes model using logistic regression is %s percent", accuracy.lr)
```

```
## [1] "The accuracy of the Naive Bayes model using logistic regression is 80.8599508599509 percent"
```

```r
#Generating the confusionMatrix for the logistic regression
confusionMatrix(as.factor(lr.prediction), prediction_labels)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  <=50K  >50K
##      <=50K   5840  1198
##      >50K     340   763
##
##                Accuracy : 0.8111
##                  95% CI : (0.8024, 0.8195)
##     No Information Rate : 0.7591
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3927
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9450
##             Specificity : 0.3891
##          Pos Pred Value : 0.8298
##          Neg Pred Value : 0.6917
##              Prevalence : 0.7591
##          Detection Rate : 0.7174
##    Detection Prevalence : 0.8645
##       Balanced Accuracy : 0.6670
##
##        'Positive' Class :  <=50K
##
```

---Problem 2---

1. Load and then explore the data set on car sales referenced by the article Shonda Kuiper (2008) Introduction to Multiple Regression: How Much Is Your Car Worth?, Journal of Statistics Education, 16:3, DOI: 10.1080/10691898.2008.11889579.

```r
#Importing liraries to read the .xlsx file
library(xlsx)
```

```
## Warning: package 'xlsx' was built under R version 4.0.2
```

```r
#Reading the kellycarsalesdata.xlsx data
cars_data <- read.xlsx("kellycarsalesdata.xlsx", sheetIndex = 1)

#Creating a copy of the cars data
cars.data <- cars_data

#Looking at the head and the structure of the data
head(cars.data)
```

```
##      Price Mileage  Make Cylinder Liter Doors Cruise Sound Leather
## 1 17314.10    8221 Buick        6   3.1     4      1     1       1
## 2 17542.04    9135 Buick        6   3.1     4      1     1       0
```

```
## 3 16218.85   13196 Buick        6   3.1     4      1     1       0
## 4 16336.91   16342 Buick        6   3.1     4      1     0       0
## 5 16339.17   19832 Buick        6   3.1     4      1     0       1
## 6 15709.05   22236 Buick        6   3.1     4      1     1       0
```

```
str(cars.data)
```

```
## 'data.frame':    804 obs. of  9 variables:
##  $ Price   : num  17314 17542 16219 16337 16339 ...
##  $ Mileage : num  8221 9135 13196 16342 19832 ...
##  $ Make    : chr  "Buick" "Buick" "Buick" "Buick" ...
##  $ Cylinder: num  6 6 6 6 6 6 6 6 6 6 ...
##  $ Liter   : num  3.1 3.1 3.1 3.1 3.1 3.1 3.1 3.1 3.1 3.1 ...
##  $ Doors   : num  4 4 4 4 4 4 4 4 4 4 ...
##  $ Cruise  : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ Sound   : num  1 1 1 0 0 1 1 1 0 1 ...
##  $ Leather : num  1 0 0 0 1 0 0 0 1 1 ...
```

```
#Summarising the cars data
summary(cars.data)
```

```
##      Price          Mileage           Make              Cylinder
##  Min.   : 8639   Min.   :  266   Length:804         Min.   :4.000
##  1st Qu.:14273   1st Qu.:14624   Class :character   1st Qu.:4.000
##  Median :18025   Median :20914   Mode  :character   Median :6.000
##  Mean   :21343   Mean   :19832                      Mean   :5.269
##  3rd Qu.:26717   3rd Qu.:25213                      3rd Qu.:6.000
##  Max.   :70755   Max.   :50387                      Max.   :8.000
##      Liter           Doors           Cruise           Sound
##  Min.   :1.600   Min.   :2.000   Min.   :0.0000   Min.   :0.0000
##  1st Qu.:2.200   1st Qu.:4.000   1st Qu.:1.0000   1st Qu.:0.0000
##  Median :2.800   Median :4.000   Median :1.0000   Median :1.0000
##  Mean   :3.037   Mean   :3.527   Mean   :0.7525   Mean   :0.6791
##  3rd Qu.:3.800   3rd Qu.:4.000   3rd Qu.:1.0000   3rd Qu.:1.0000
##  Max.   :6.000   Max.   :4.000   Max.   :1.0000   Max.   :1.0000
##     Leather
##  Min.   :0.0000
##  1st Qu.:0.0000
##  Median :1.0000
##  Mean   :0.7239
##  3rd Qu.:1.0000
##  Max.   :1.0000
```

```
#Factorising the cars data and make of the  the car
cars.data$Make <- as.factor(cars.data$Make)
```

2. Are there outliers in the data set? How do you identify outliers and how do you deal with them? Remove them but create a second data set with outliers removed. Keep the original data set. -> Outlier are identified using z-score method, we impute them using the mean method

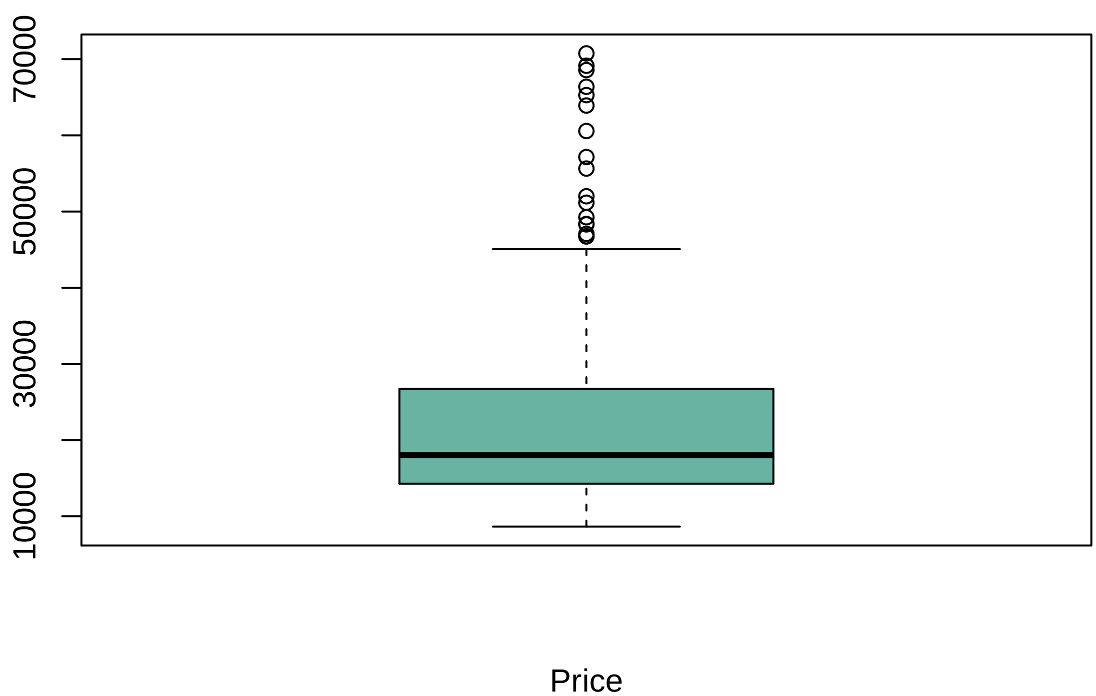
```
#Importing the libraries to plot BarPlot and caluclate outliers
library(tidyr)
library(hrbrthemes)
```

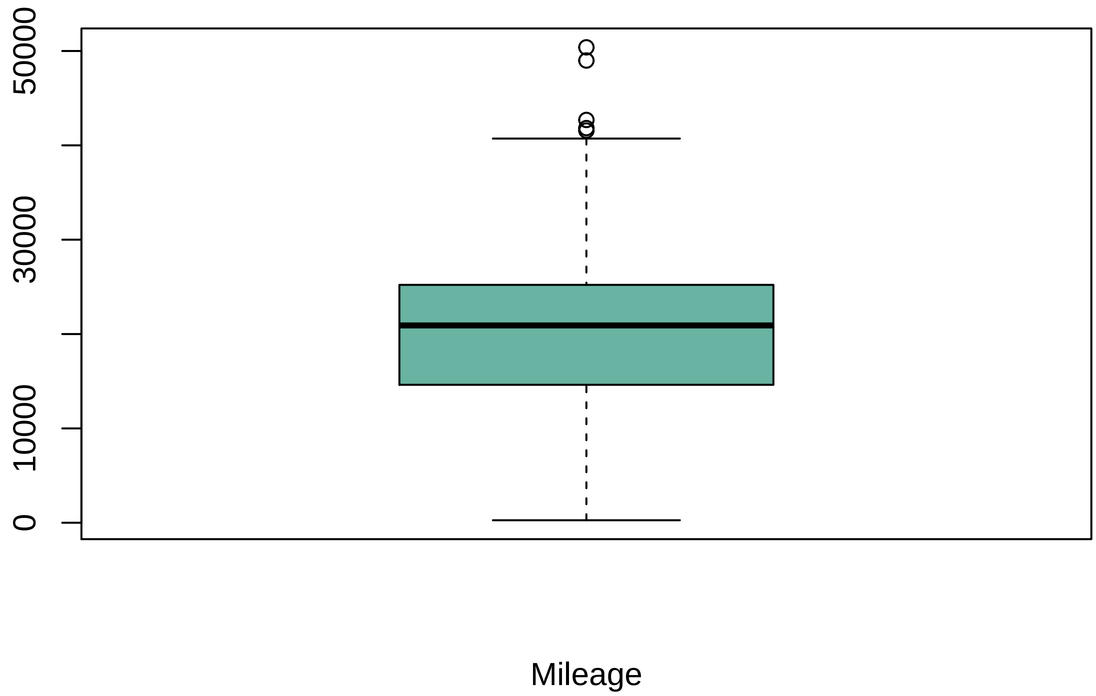
```
## Warning: package 'hrbrthemes' was built under R version 4.0.2

## NOTE: Either Arial Narrow or Roboto Condensed fonts are required to use these themes.

##       Please use hrbrthemes::import_roboto_condensed() to install Roboto Condensed and

##       if Arial Narrow is not on your system, please see https://bit.ly/arialnarrow
```

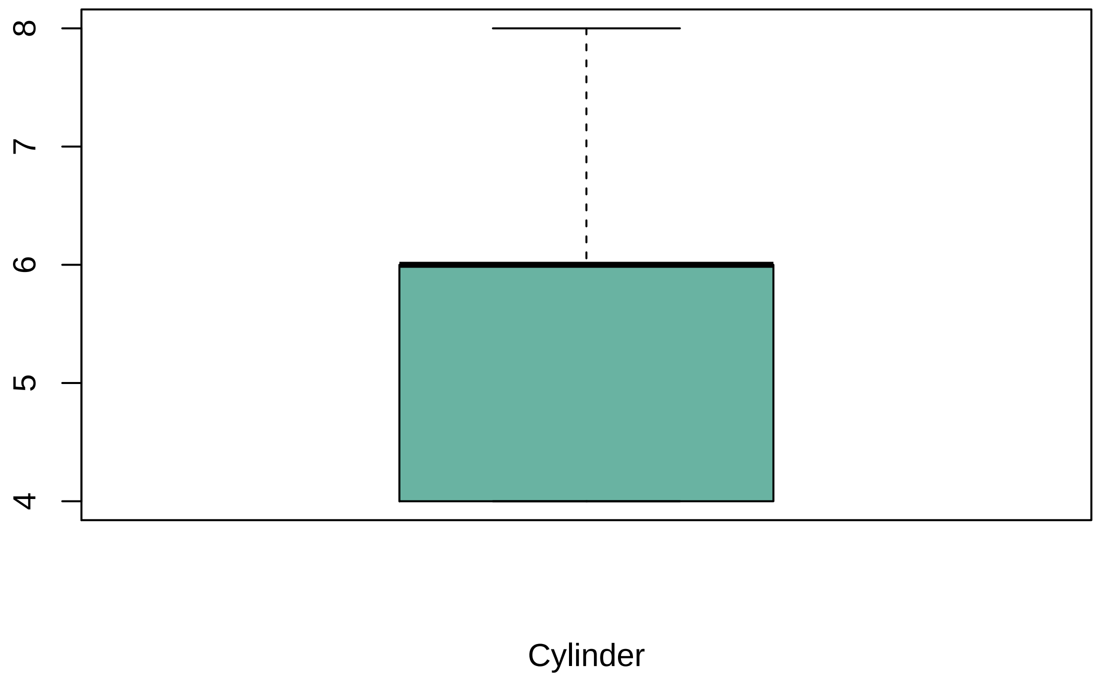
```
library(viridis)
```

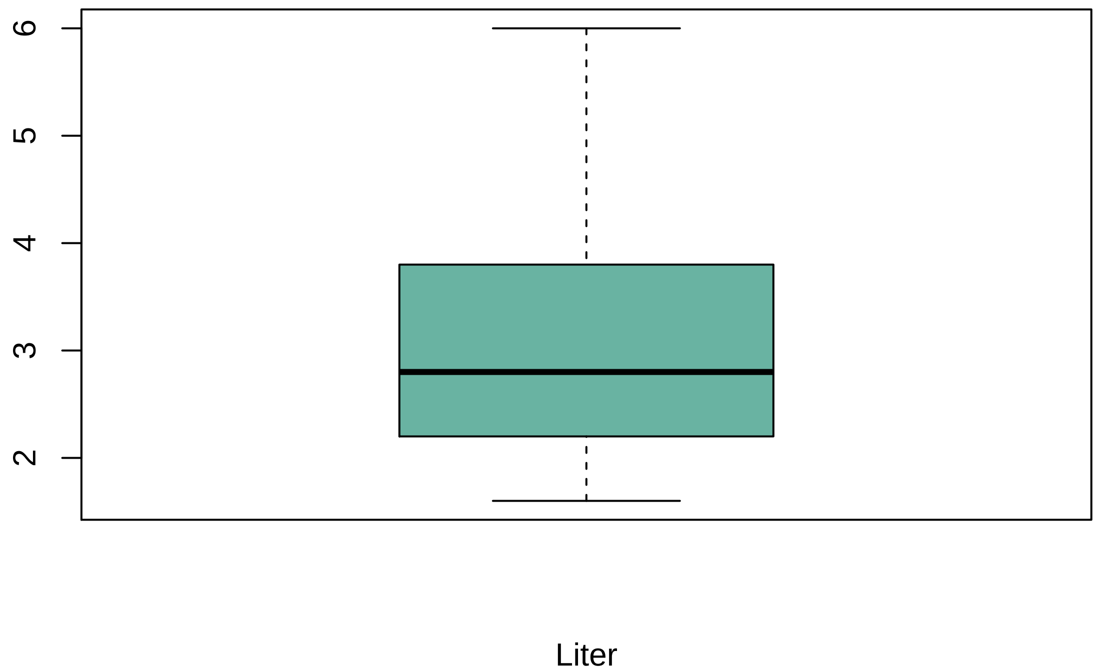
```
## Warning: package 'viridis' was built under R version 4.0.2

## Loading required package: viridisLite
```

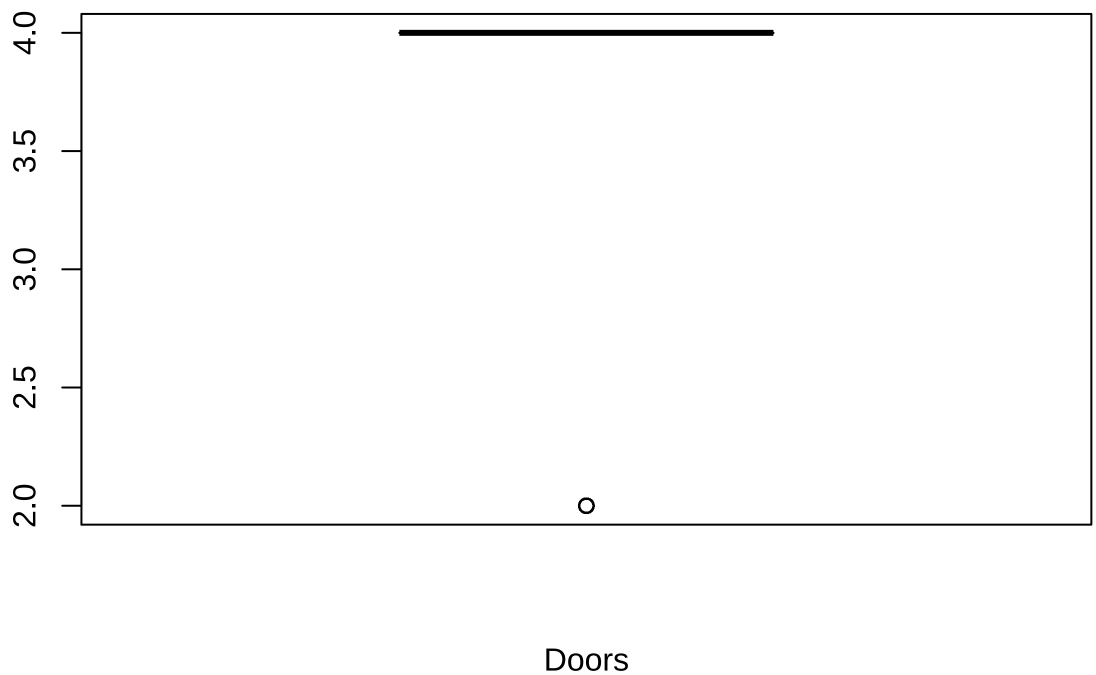
```
library(outliers)


#Using for loop to plot BarPlot for all the Variables
for (i in c("Price", "Mileage", "Cylinder", "Liter", "Doors", "Cruise", "Sound", "Leather"))
{
  boxplot(cars.data[,i], col = "#69B3A2", xlab = i)
}
```

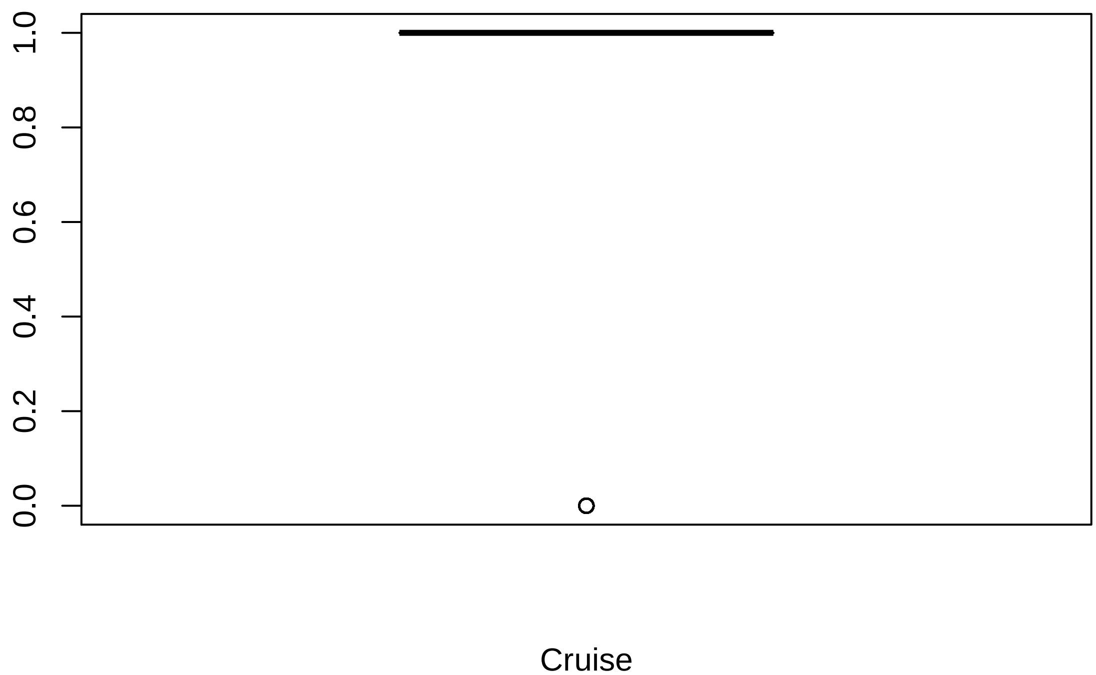


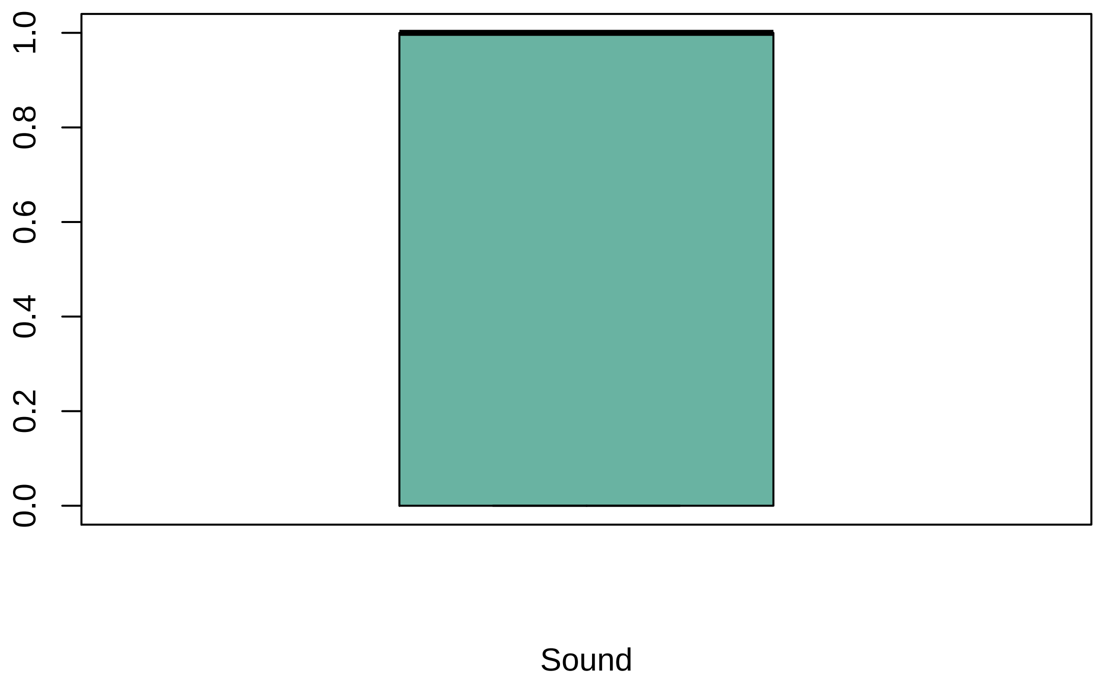

0
10000
30000
50000
Mileage

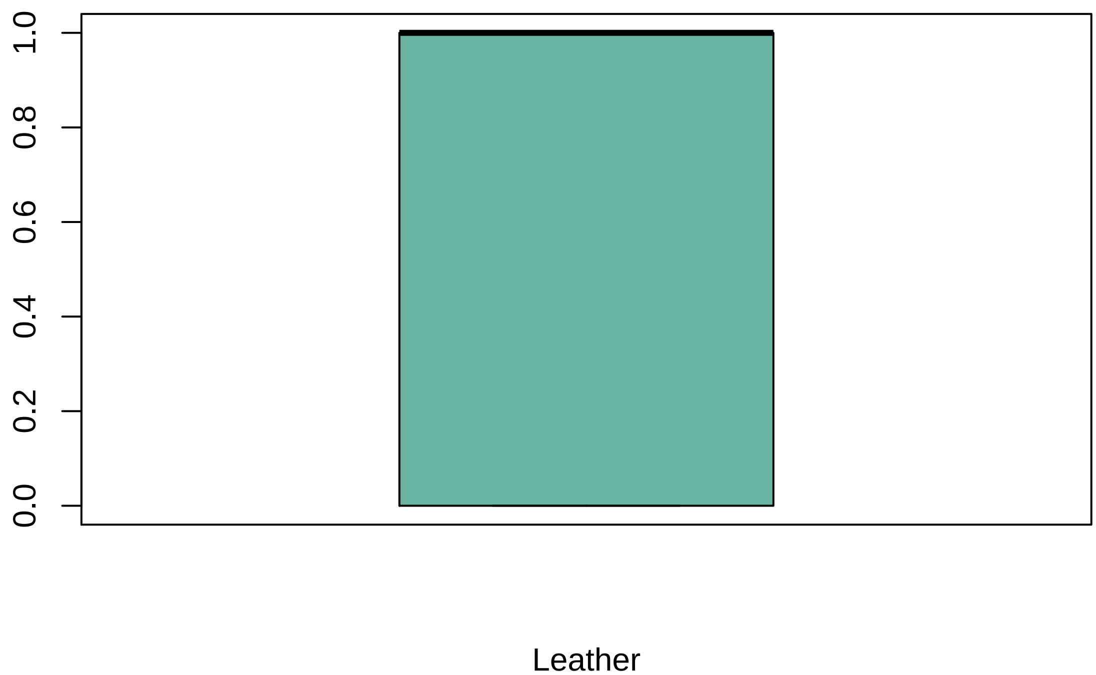

8
7
6
5
4
Cylinder


6
5
4
3
2
Liter


4.0
3.5
3.0
2.5
2.0
Doors


1.0
0.8
0.6
0.4
0.2
0.0
Cruise


1.0
0.8
0.6
0.4
0.2
0.0
Sound




```r
#Creating a copy of the car data to eliminate outliers
cars.outlier <- cars_data

#Using for loop and scores() function to caluclate outlier and replace them with NA
for (i in c("Price", "Mileage", "Cylinder", "Liter", "Doors", "Cruise", "Sound", "Leather"))
{
  zscore <- abs(scores(cars.outlier[,i]))
  cars.outlier[which((zscore > 3)),i] = NA
}

#Checking for outliers in the data
anyNA(cars.outlier)
```

```
## [1] TRUE
```

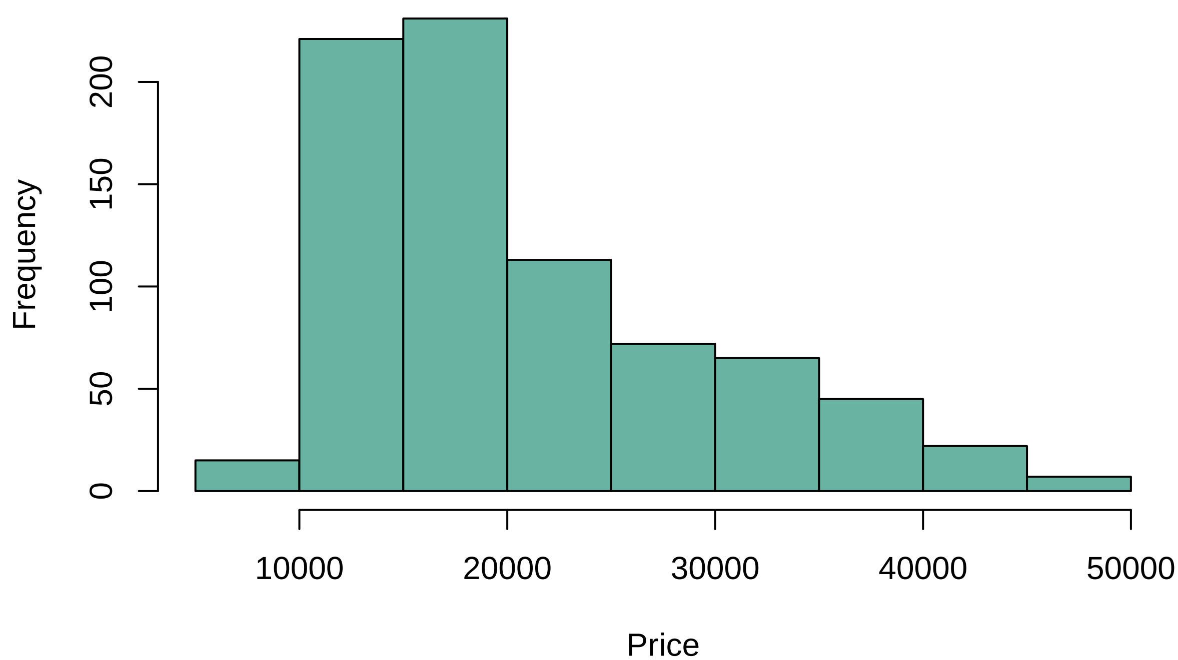
```r
#Dropping the NA values from the data
new_cars.data <- cars.outlier %>% drop_na()

#Verifing wheather all NA values are removed from the data
anyNA(new_cars.data)
```

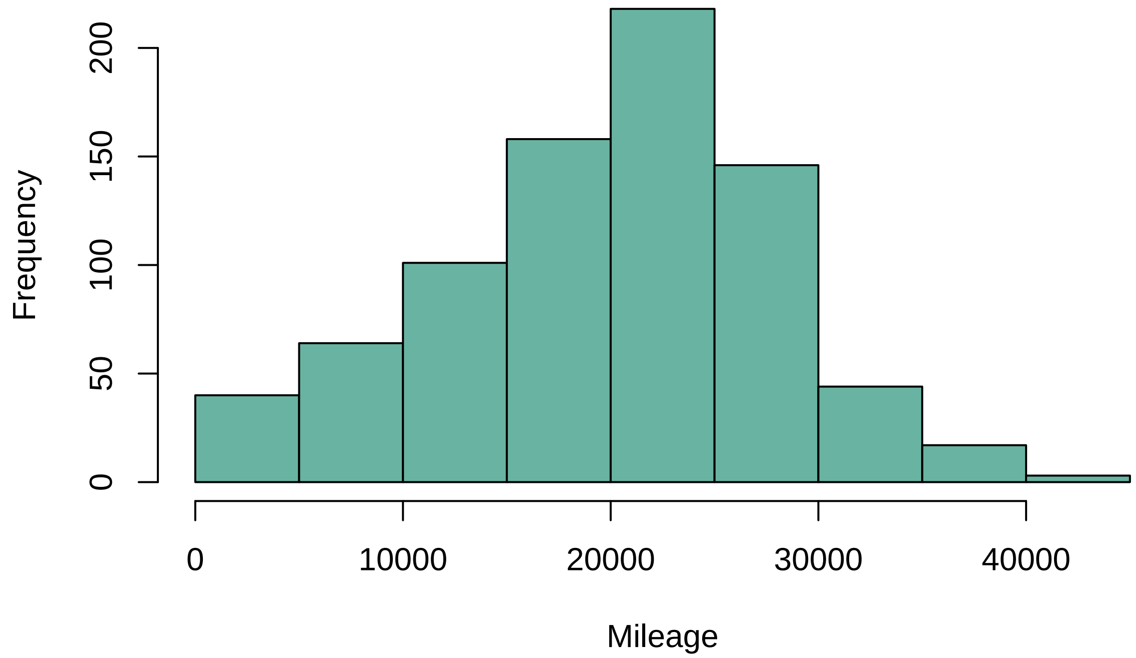
```
## [1] FALSE
```

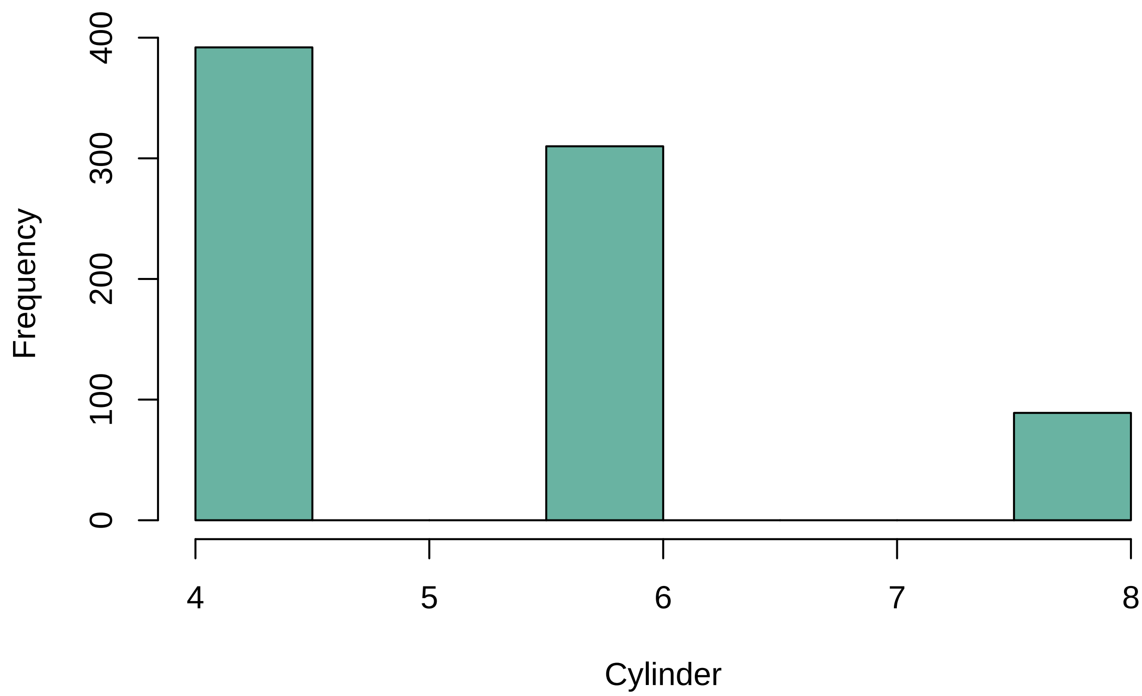
3. What are the distributions of each of the features in the data set with outliers removed? Are they reasonably normal so you can apply a statistical learner such as regression? Can you normalize features

through a log, inverse, or square-root transform? Transform as needed. -> Out of all features only Mileage is normally distributed. Price and Liter have skewed distributions. Rest features are categorical so the distribution do not matter. Hence we transform only Price and liter. But during transformation I observed that Liter does not change even with transformation so we neglect the transformation of Liter. Meanwhile Price becomes fairly normal after log transformation.

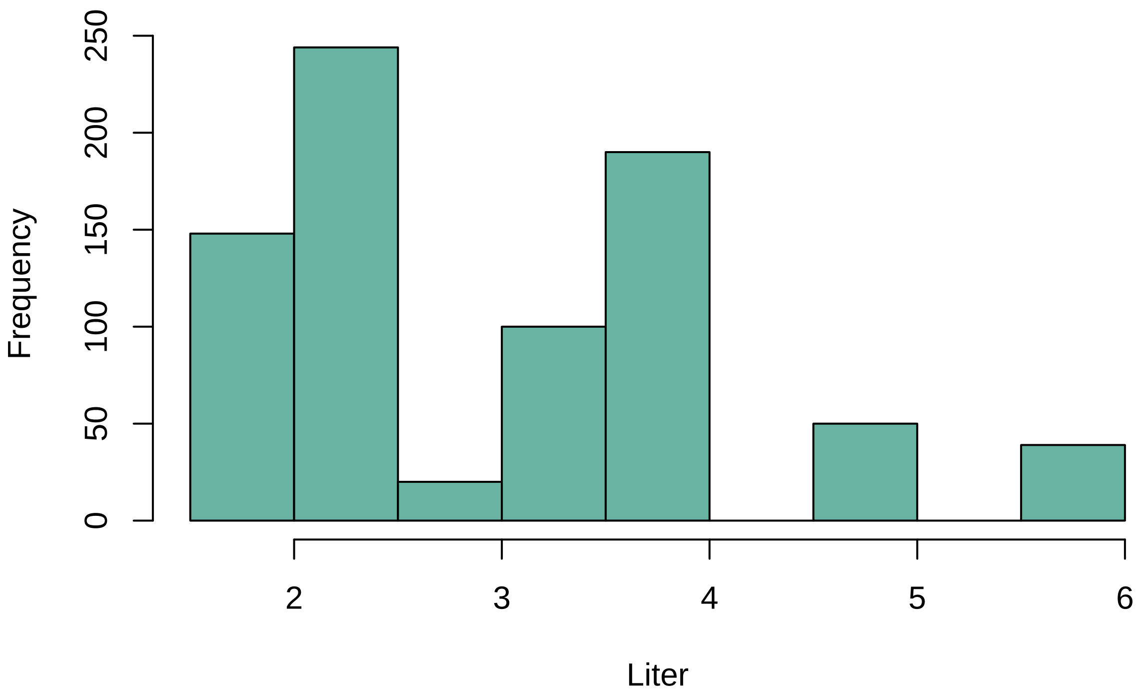
```
#Importing library to plot the Histogram of the cars data
library(ggplot2)

#Using for loop to plot the histogram of the cars data
for (i in c("Price", "Mileage", "Cylinder", "Liter", "Doors", "Cruise", "Sound", "Leather"))
{
hist(new_cars.data[,i], col = "#69B3A2", main = NULL, xlab = i)
}
```



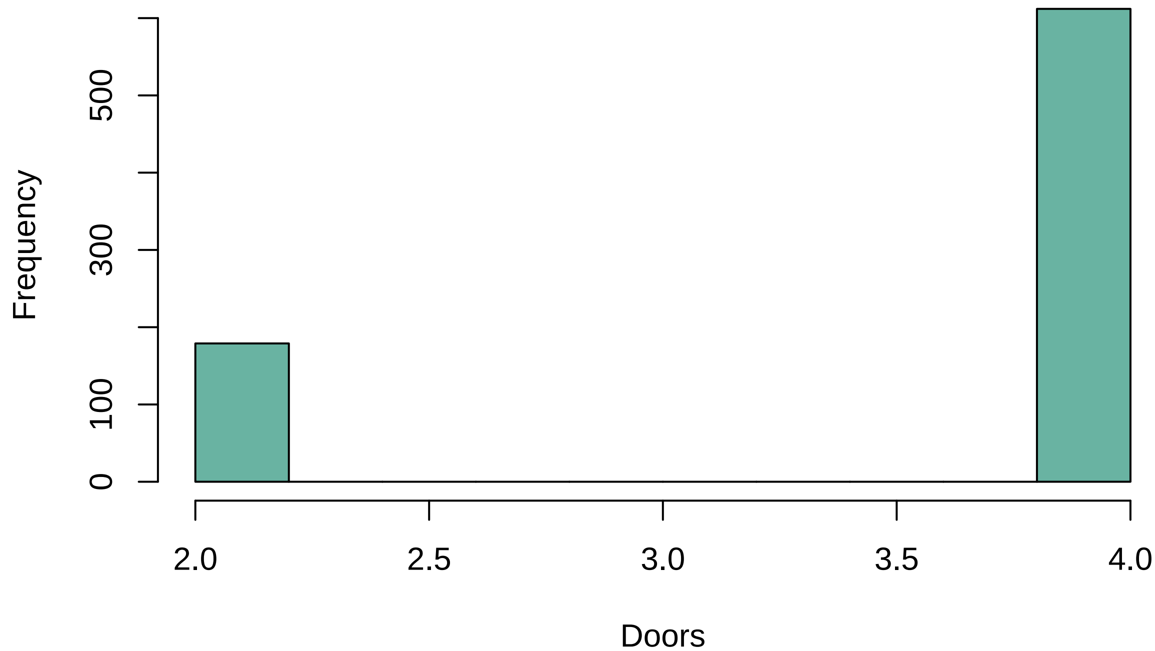

Frequency
0
50
100
150
200
0
10000
20000
30000
40000
Mileage

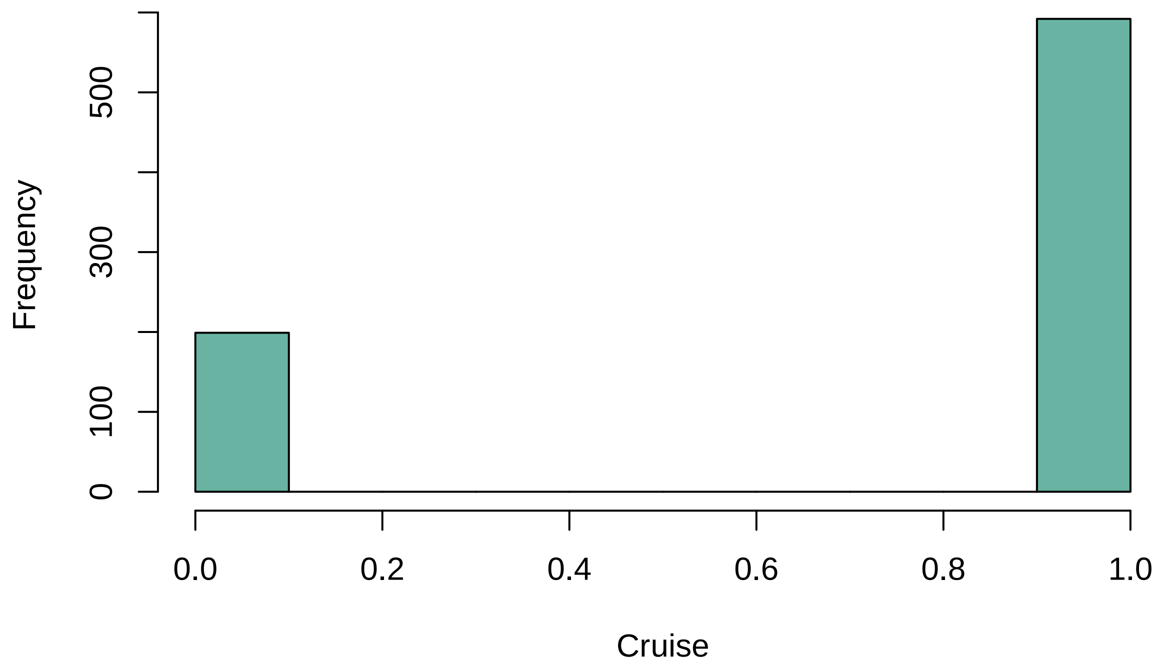


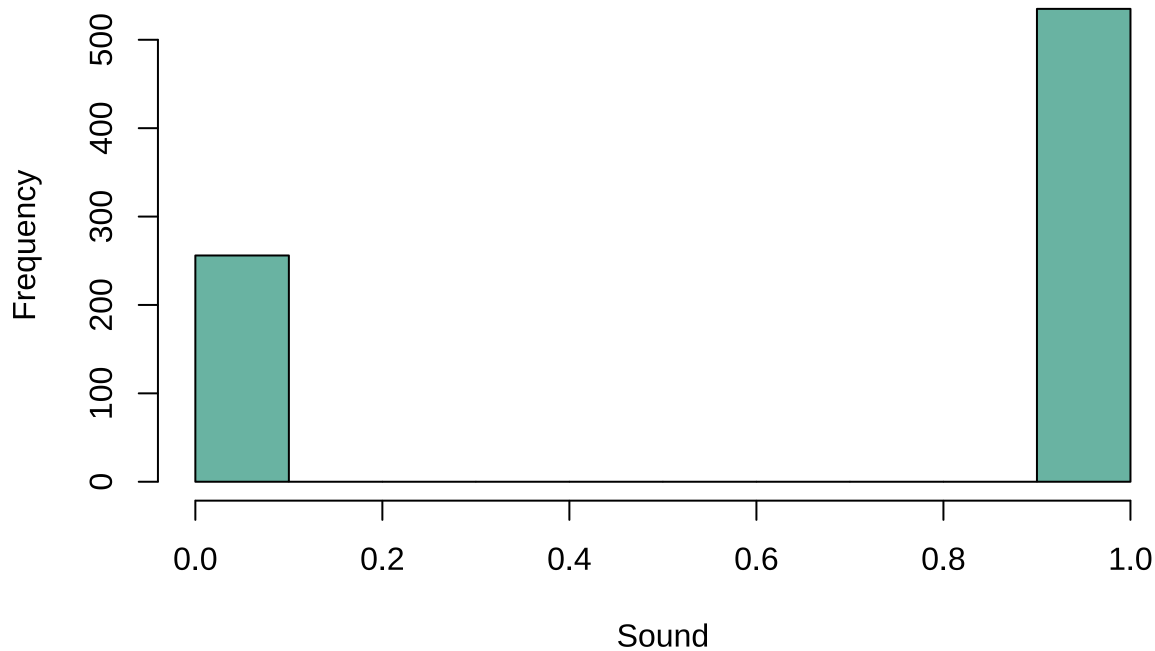
Frequency
0
100
200
300
400
4
5
6
7
8
Cylinder

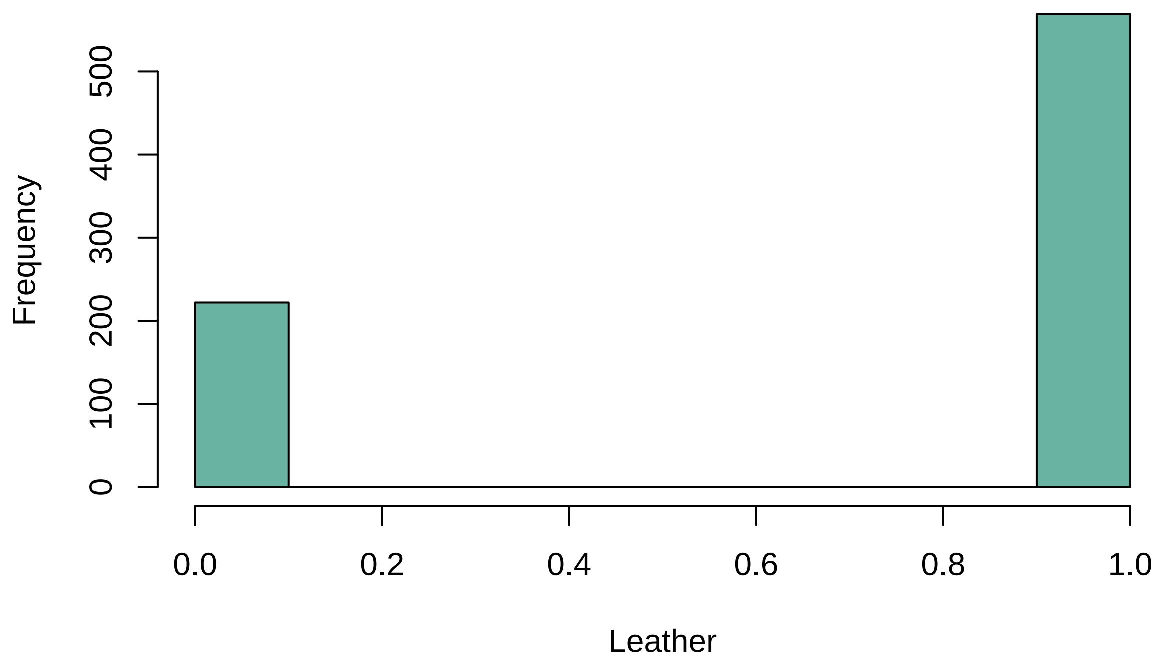


Frequency
0
50
100
150
200
250
2
3
4
5
6
Liter

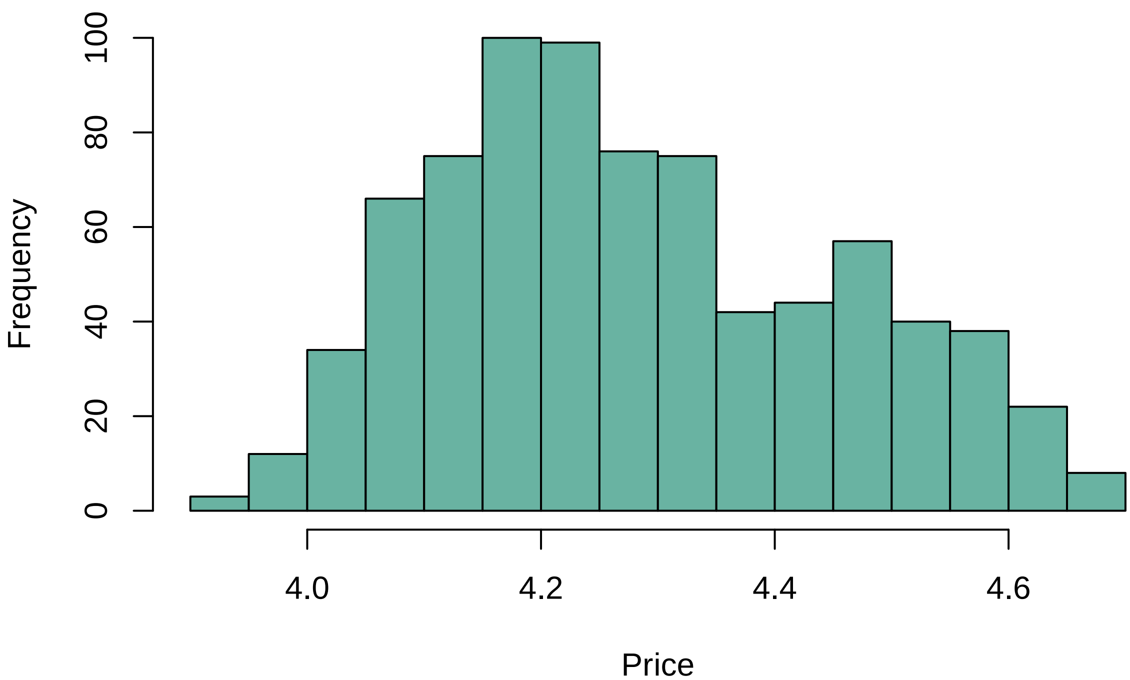

Frequency
0
100
300
500
2.0
2.5
3.0
3.5
4.0
Doors

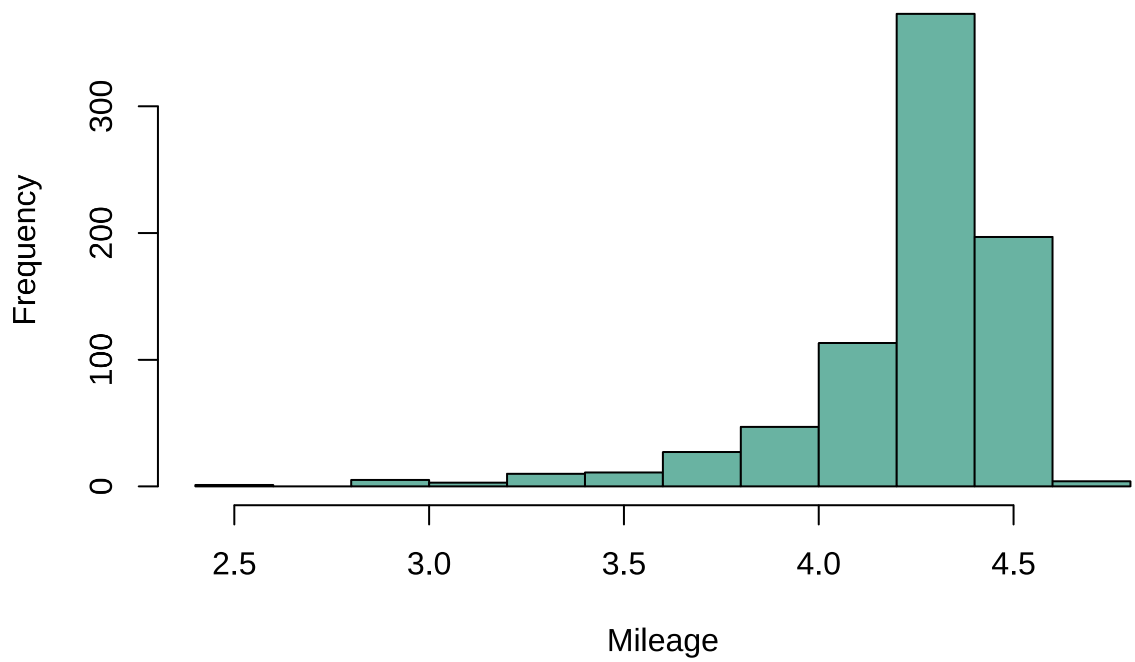

Frequency
0
100
300
500
0.0
0.2
0.4
0.6
0.8
1.0
Cruise

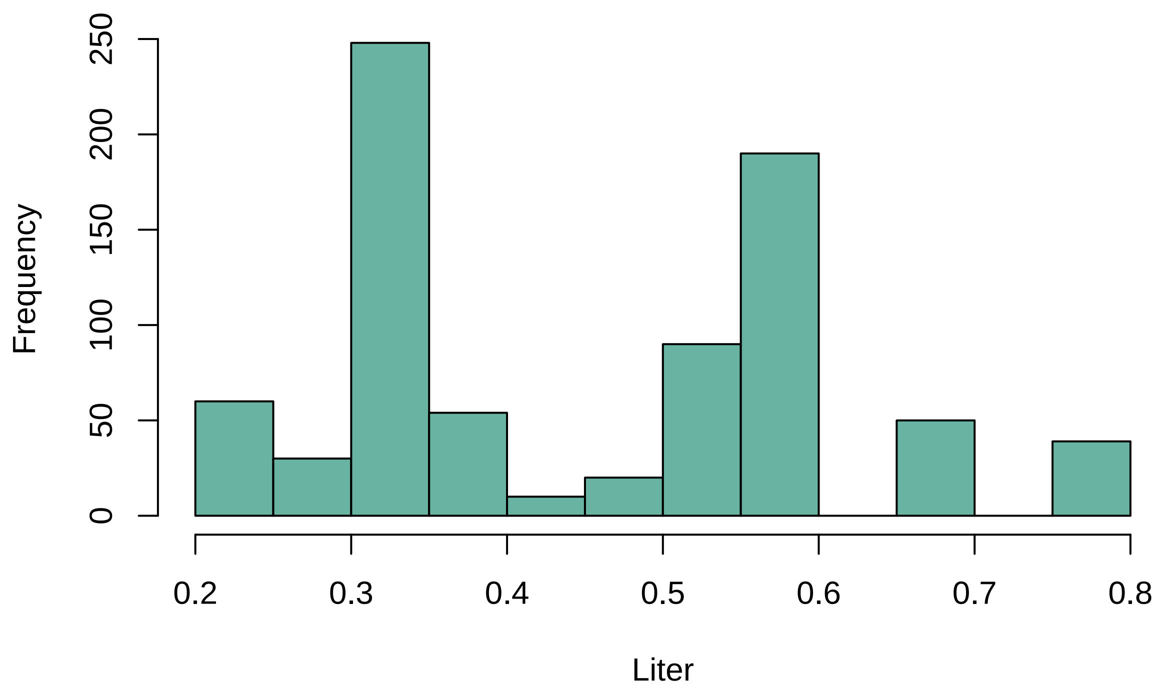

Frequency
0
100
200
300
400
500
0.0
0.2
0.4
0.6
0.8
1.0
Sound




```r
#Using for loop to plot the histogram for the cars data with log transformed
for (i in c("Price", "Mileage", "Liter"))
{
hist(log10(new_cars.data[,i]), col = "#69B3A2", main = NULL, xlab = i)
}
```


Frequency
0
20
40
60
80
100
4.0
4.2
4.4
4.6
Price


Frequency
0
100
200
300
2.5
3.0
3.5
4.0
4.5
Mileage




```r
#Log transforming the Price value in the cars data
new_cars.data$Price <- log10(new_cars.data$Price)
```

4. What are the correlations to the response variable (car sales price) and are there collinearities? Build a full correlation matrix. -> Based on the observation from the correlation matrix, the meaningful insight which is observed is that Cylinder, Liter, and Cruise are highly correlated to the Price column.

```r
#Correlation matrix of price vs other features
cor(cars.data["Price"],cars.data[c("Mileage", "Cylinder", "Liter", "Doors", "Cruise", "Sound", "Leather
```

```
##          Mileage  Cylinder     Liter      Doors    Cruise      Sound   Leather
## Price -0.1430505 0.5690861 0.5581458 -0.1387497 0.4308515 -0.1243478 0.1571969
```

```r
#Full correlation matrix
cor(cars.data[c("Price", "Mileage", "Cylinder", "Liter", "Doors", "Cruise", "Sound", "Leather")])
```

```
##               Price      Mileage    Cylinder       Liter       Doors
## Price     1.0000000 -0.143050506  0.56908614  0.55814581 -0.13874965
## Mileage  -0.1430505  1.000000000 -0.02946099 -0.01864062 -0.01694449
## Cylinder  0.5690861 -0.029460989  1.00000000  0.95789658  0.00220592
## Liter     0.5581458 -0.018640622  0.95789658  1.00000000 -0.07925909
## Doors    -0.1387497 -0.016944490  0.00220592 -0.07925909  1.00000000
## Cruise    0.4308515  0.025036652  0.35428485  0.37750927 -0.04767418
## Sound    -0.1243478 -0.026145926 -0.08970430 -0.06552707 -0.06253031
```

```
## Leather   0.1571969  0.001005446  0.07551962  0.08733194 -0.06196858
##               Cruise       Sound      Leather
## Price     0.43085149 -0.12434785  0.157196855
## Mileage   0.02503665 -0.02614593  0.001005446
## Cylinder  0.35428485 -0.08970430  0.075519616
## Liter     0.37750927 -0.06552707  0.087331945
## Doors    -0.04767418 -0.06253031 -0.061968579
## Cruise    1.00000000 -0.09173015 -0.070573094
## Sound    -0.09173015  1.00000000  0.165443625
## Leather  -0.07057309  0.16544362  1.000000000
```

5. Split the data set 75/25 so you retain 25% for testing using random sampling.

```
#Setting the seed for sampling to 1
set.seed(1)

#Spliting the dataset using createDataPartition() function, so that the data is samped in equal distrib
cars.sample <- createDataPartition(cars.data$Make, p = 0.25, list = FALSE, times = 1)

#Assigining 25% of the car data for testing and the remaining 75% for training
cars.testing <- cars.data[cars.sample,]
cars.training <- cars.data[-cars.sample,]

#Spliting the dataset using createDataPartition() function, so that the data is samped in equal distrib
new_cars.sample <- createDataPartition(new_cars.data$Make, p = 0.25, list = FALSE, times = 1)

#Assigining 25% of the new car data for testing and the remaining 75% for training
new_cars.testing <- new_cars.data[new_cars.sample,]
new_cars.training <- new_cars.data[-new_cars.sample,]
```

6. Build a full multiple regression model for predicting car sales prices in this data set using the complete training data set (no outliers removed), i.e., a regression model that contains all features regardless of their p-values.

```
#Building a multiple logistic regression fro predicting car sales price
cars.lr <- lm(Price~., data = cars.training)

#Summarising the model
summary(cars.lr)
```

```
##
## Call:
## lm(formula = Price ~ ., data = cars.training)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -9267.2 -1929.6  -179.9  1329.5 22853.7
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)  15471.2521  1464.7671  10.562  < 2e-16 ***
## Mileage         -0.1939     0.0174 -11.145  < 2e-16 ***
```

```
## MakeCadillac  16178.5653   750.4163  21.559  < 2e-16 ***
## MakeChevrolet -2027.5782   554.0816  -3.659 0.000276 ***
## MakePontiac   -1944.5090   567.8856  -3.424 0.000660 ***
## MakeSAAB      14838.4667   674.1391  22.011  < 2e-16 ***
## MakeSaturn    -2169.1919   745.9068  -2.908 0.003773 **
## Cylinder       -369.4893   456.9854  -0.809 0.419108
## Liter          4993.2047   519.6213   9.609  < 2e-16 ***
## Doors         -1565.4533   175.2982  -8.930  < 2e-16 ***
## Cruise         -289.8027   392.5100  -0.738 0.460607
## Sound           -64.9374   313.1588  -0.207 0.835798
## Leather        -162.1975   342.3986  -0.474 0.635883
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3408 on 589 degrees of freedom
## Multiple R-squared:  0.8827, Adjusted R-squared:  0.8803
## F-statistic: 369.3 on 12 and 589 DF,  p-value: < 2.2e-16
```

7. Build an ideal multiple regression model using backward elimination based on p-value for predicting car sales prices in this data set using the complete training data set with outliers removed (Question 2) and features transformed (Question 3). Provide a detailed analysis of the model using the training data set with outliers removed and features transformed, including Adjusted R-Squared, RMSE, and p-values of all coefficients.

```r
#Importing laibrary for feature transformation
library(SignifReg)

#Building a multiple logistic regression fro predicting car sales price with outliers removed
new_cars.lr <- lm(Price~., data = new_cars.training)

#Summarising the model for which ouliers are removed
summary(new_cars.lr)
```

```
##
## Call:
## lm(formula = Price ~ ., data = new_cars.training)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.125712 -0.029655  0.000341  0.025157  0.133968
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)    4.108e+00  2.001e-02 205.271  < 2e-16 ***
## Mileage       -3.465e-06  2.395e-07 -14.468  < 2e-16 ***
## MakeCadillac   1.868e-01  1.082e-02  17.254  < 2e-16 ***
## MakeChevrolet -6.330e-02  7.509e-03  -8.430 2.75e-16 ***
## MakePontiac   -3.648e-02  7.624e-03  -4.785 2.17e-06 ***
## MakeSAAB       2.889e-01  9.126e-03  31.654  < 2e-16 ***
## MakeSaturn    -5.749e-02  1.015e-02  -5.666 2.31e-08 ***
## Cylinder      -1.073e-02  6.591e-03  -1.628   0.1041
## Liter          1.086e-01  7.389e-03  14.704  < 2e-16 ***
## Doors         -1.254e-02  2.473e-03  -5.072 5.30e-07 ***
```

```
## Cruise         -5.204e-04  5.389e-03  -0.097   0.9231
## Sound          -7.563e-03  4.281e-03  -1.767   0.0778 .
## Leather         1.267e-03  4.587e-03   0.276   0.7825
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.04598 on 579 degrees of freedom
## Multiple R-squared:  0.9278, Adjusted R-squared:  0.9263
## F-statistic: 619.8 on 12 and 579 DF,  p-value: < 2.2e-16
```

```
#Performing backward elimination using p-value
new_cars.lr <- drop1SignifReg(new_cars.lr,alpha = 0.05,criterion="p-value")
```

```
##            RSS     AIC         BIC         adj.rsq PRESS   max_pvalue max VIF
## - Liter    1.6813  -1765.43057 -1708.44498 0.89892 1.75639 0.32703    5
## - Doors    1.27857 -1927.53545 -1870.54986 0.92313 1.33665 0.77546    21.27578
## - Cruise   1.22419 -1953.26428 -1896.27869 0.9264  1.27993 0.7755     22.80918
## - Cylinder 1.22977 -1950.57104 -1893.58545 0.92606 1.28551 0.89008    5
## - Leather  1.22433 -1953.1958  -1896.21021 0.92639 1.27903 0.9048     22.52342
## <none>     1.22417 -1951.27381 -1889.90472 0.92627 1.28409 0.9231     22.82426
## - Sound    1.23077 -1950.09107 -1893.10549 0.926   1.2858  0.95285    22.70316
## - Mileage  1.66673 -1770.58223 -1713.59665 0.89979 1.74234 0.98095    22.80694
##            alpha_cut-off Bonferroni FDR
## - Liter    FALSE         FALSE      FALSE
## - Doors    FALSE         FALSE      FALSE
## - Cruise   FALSE         FALSE      FALSE
## - Cylinder FALSE         FALSE      FALSE
## - Leather  FALSE         FALSE      FALSE
## <none>     FALSE         FALSE      FALSE
## - Sound    FALSE         FALSE      FALSE
## - Mileage  FALSE         FALSE      FALSE
```

```
summary(new_cars.lr)
```

```
##
## Call:
## lm(formula = Price ~ Mileage + Make + Cylinder + Doors + Cruise +
##     Sound + Leather, data = new_cars.training)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.144595 -0.033159  0.000422  0.031182  0.158114
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)     3.980e+00  2.109e-02 188.705  < 2e-16 ***
## Mileage        -3.356e-06  2.803e-07 -11.976  < 2e-16 ***
## MakeCadillac    1.165e-01  1.137e-02  10.246  < 2e-16 ***
## MakeChevrolet -6.561e-02  8.791e-03  -7.464 3.10e-13 ***
## MakePontiac    -4.096e-02  8.921e-03  -4.591 5.40e-06 ***
## MakeSAAB        2.993e-01  1.065e-02  28.095  < 2e-16 ***
## MakeSaturn     -5.080e-02  1.187e-02  -4.280 2.19e-05 ***
## Cylinder        8.144e-02  2.386e-03  34.130  < 2e-16 ***
```

```
## Doors         -2.313e-02  2.770e-03  -8.352 4.97e-16 ***
## Cruise         6.829e-03  6.283e-03   1.087   0.2775
## Sound         -4.913e-03  5.008e-03  -0.981   0.3270
## Leather        1.076e-02  5.317e-03   2.023   0.0435 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.05384 on 580 degrees of freedom
## Multiple R-squared:  0.9008, Adjusted R-squared:  0.8989
## F-statistic: 478.8 on 11 and 580 DF,  p-value: < 2.2e-16
```

```
new_cars.lr <- drop1SignifReg(new_cars.lr,alpha = 0.05,criterion="p-value")
```

```
##            RSS     AIC         BIC         adj.rsq PRESS   max_pvalue max VIF
## - Cylinder 5.05798 -1115.40066 -1062.79858 0.69643 5.22323 0.01691    5
## - Sound    1.68408 -1766.44924 -1713.84716 0.89892 1.75239 0.26996    5
## - Cruise   1.68472 -1766.2257  -1713.62363 0.89889 1.75409 0.31785    5
## <none>     1.6813  -1765.43057 -1708.44498 0.89892 1.75639 0.32703    5
## - Doors    1.88348 -1700.20446 -1647.60238 0.88696 1.96238 0.38763    5
## - Leather  1.69316 -1763.26753 -1710.66545 0.89838 1.76241 0.4701     5
## - Mileage  2.09704 -1636.62207 -1584.01999 0.87414 2.18188 0.48868    5
##            alpha_cut-off Bonferroni FDR
## - Cylinder TRUE          FALSE      TRUE
## - Sound    FALSE         FALSE      FALSE
## - Cruise   FALSE         FALSE      FALSE
## <none>     FALSE         FALSE      FALSE
## - Doors    FALSE         FALSE      FALSE
## - Leather  FALSE         FALSE      FALSE
## - Mileage  FALSE         FALSE      FALSE
```

```
summary(new_cars.lr)
```

```
##
## Call:
## lm(formula = Price ~ Mileage + Make + Doors + Cruise + Sound +
##     Leather, data = new_cars.training)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.26210 -0.05403  0.00572  0.04160  0.35931
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)     4.417e+00  2.905e-02 152.063  < 2e-16 ***
## Mileage        -3.305e-06  4.857e-07  -6.804 2.54e-11 ***
## MakeCadillac    2.337e-01  1.879e-02  12.436  < 2e-16 ***
## MakeChevrolet -1.144e-01  1.503e-02  -7.613 1.09e-13 ***
## MakePontiac   -6.259e-02  1.542e-02  -4.059 5.61e-05 ***
## MakeSAAB       1.285e-01  1.630e-02   7.886 1.55e-14 ***
## MakeSaturn    -1.539e-01  1.989e-02  -7.739 4.47e-14 ***
## Doors         -2.841e-02  4.792e-03  -5.929 5.24e-09 ***
## Cruise         8.618e-02  1.011e-02   8.520  < 2e-16 ***
## Sound         -2.306e-02  8.630e-03  -2.672  0.00775 **
```

```
## Leather        2.203e-02  9.196e-03   2.395  0.01691 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.0933 on 581 degrees of freedom
## Multiple R-squared:  0.7016, Adjusted R-squared:  0.6964
## F-statistic: 136.6 on 10 and 581 DF,  p-value: < 2.2e-16
```

```
#Calculating the root mean squared error
rmse <- sqrt(mean(new_cars.lr$residuals^2))
sprintf("rmse for the model is %s",rmse)
```

```
## [1] "rmse for the model is 0.0924330968865731"
```

```
#Calculating the adjusted R-Squared value
sprintf("rmse for the model is 0.6964")
```

```
## [1] "rmse for the model is 0.6964"
```

8. On average, by how much do we expect a leather interior to change the resale value of a car based on the models built in (6) and in (7)? Note that 1 indicates the presence of leather in the car. -> Based on calculation it is seen that if the leather is present in the car model, then Price is affected by 162 USD

```
#Calculating price when leather is present
leather_interior_present <- cars.lr$coefficients["(Intercept)"] +
                             cars.lr$coefficients["Mileage"] * mean(cars.training$Mileage) +
                             cars.lr$coefficients["Cylinder"] * mean(cars.training$Cylinder) +
                             cars.lr$coefficients["Liter"] * mean(cars.training$Liter) +
                             cars.lr$coefficients["Doors"] * mean(cars.training$Doors) +
                             cars.lr$coefficients["Cruise"] * mean(cars.training$Cruise) +
                             cars.lr$coefficients["Sound"] * mean(cars.training$Sound) +
                             cars.lr$coefficients["Leather"] * 1

#Calculating price when leather is absent
leather_interior_absent <- cars.lr$coefficients["(Intercept)"] +
                            cars.lr$coefficients["Mileage"] * mean(cars.training$Mileage) +
                            cars.lr$coefficients["Cylinder"] * mean(cars.training$Cylinder) +
                            cars.lr$coefficients["Liter"] * mean(cars.training$Liter) +
                            cars.lr$coefficients["Doors"] * mean(cars.training$Doors) +
                            cars.lr$coefficients["Cruise"] * mean(cars.training$Cruise) +
                            cars.lr$coefficients["Sound"] * mean(cars.training$Sound) +
                            cars.lr$coefficients["Leather"] * 0

#Printing the change in resale value
change_in_price <- abs(leather_interior_present - leather_interior_absent)
sprintf("The change in the resale value of the car in case of leather interior is present or absent is %
```

```
## [1] "The change in the resale value of the car in case of leather interior is present or absent is 1
```

9. Using the regression models of (6) and (7) what are the predicted resale prices of a 2005 4-door Saab with 61,435 miles with a leather interior, a 4-cylinder 2.3 liter engine, cruise control, and a premium sound

system? Why are the predictions different? -> Prediction values are different because the accuracy of both models varies. Apart from that, the problem 2.6 model has all features included because of which the RMSE of the model is high. In contrast, the problem 2.7 model has only significant features selected, so the RMSE is low, and the accuracy is high.

```
#Creating a new data from the sample car data for testing
new_car <- data.frame(NA, 61435, "SAAB", 4, 2.3, 4, 1, 1, 1)

#Adding column names to the sample car data
new_car.col_names <- c("Price", "Mileage", "Make", "Cylinder", "Liter", "Doors", "Cruise", "Sound", "Lea
colnames(new_car) <- new_car.col_names

#Creating another sample of the cars data to perform factor operation on the new adult data
new_cars.data <- cars_data[c("Price", "Mileage", "Make", "Cylinder", "Liter", "Doors", "Cruise", "Sound

#Binding the row of the new car data with the original car data
new_cars.data <- rbind(new_car, new_cars.data)

#Converting to factor the make of the car
new_cars.data$Make <- as.factor(new_cars.data$Make)

#Retriving the new car data after converting to factor
new_car_testing.data <- new_cars.data[1,]

#Testing the model for the new car data
car.pred <- predict(cars.lr, new_car_testing.data)

#Printing the values generated from the problem 2.6 model
car_pred <- unname(car.pred)
sprintf("The prediction Price for the test case by using problem 2.6 model  %s",car_pred)
```

```
## [1] "The prediction Price for the test case by using problem 2.6 model  21623.0787680458"
```

```
#Testing the model for the new car data
new_car.pred <- predict(new_cars.lr, new_car_testing.data)

#Printing the values generated from the problem 2.6 model
new_car_pred <- 10^(unname(new_car.pred))
sprintf("The prediction Price for the test case by using problem 2.7 model  %s",new_car_pred)
```

```
## [1] "The prediction Price for the test case by using problem 2.7 model  20597.8059415508"
```

10. For the regression model of (7), calculate the 95% prediction interval for the car in (9).

```
#Calculating 95% CI using interval function in predict() for problem 2.6 model
car.pred.CI <- predict(cars.lr, new_car_testing.data, interval = "confidence")
car.pred.CI <- data.frame("Predicted value" = car.pred.CI[1], "Lower Bound" = car.pred.CI[2], "Upper Bo
car.pred.CI
```

```
##   Predicted.value Lower.Bound Upper.Bound
## 1        21623.08    19991.84    23254.32
```

```r
#Calculating 95% CI using interval function in predict() for problem 2.7 model
new_car.pred.CI <- predict(new_cars.lr, new_car_testing.data, interval = "confidence")
new_car.pred.CI <- data.frame("Predicted value" = 10^new_car.pred.CI[1], "Lower Bound" = 10^new_car.pred
new_car.pred.CI
```

```
##   Predicted.value Lower.Bound Upper.Bound
## 1        20597.81    18563.97    22854.47
```