Car Evaluation Dataset - CM3111

Muhammed Umair Iqbal

December 12, 2017

1 Data Exploration

1.1 Dataset Choice

I have selected a data set that is related to cars as I have a passion for cars and thought that it would be a good exploring a data set related to cars via R Studio. I got my dataset from the UCI Machine Learning Repository from "http://archive.ics.uci.edu/ml/datasets/Car+Evaluation".

1.2 Technology Platform

The application I have used to explore my data is R Studio and carry out experiments on. After researching hadoop i have found that i do not need to be using it. For many reasons, one of them being that Hadoop can store massive amounts of data, Petabytes to be exact. Moreover Hadoop should be used when the users requires thousands of operations per second on terabytes of data at a time, which is not what is required in this instance which is why i haven't incorporated Hadoop or any other Big Data Technologies.

1.3 Problem Statement

The dataset I have choosen is a Car Evaluation dataset.It has 6 Attributes;

- Price
- Maint(Maintence of the car)
- Doors(Number of Doors)
- Persons(Number of people the car can seat at once)
- Boot(The size of the boot)
- Safety(The safety rating of the car)

The aim is to build a predictive model, to predict what the Evaluation is of the car.

1.4 Data Exploration

I began to explore my Data set, by finding out the numbers of rows and coloumns

```
#Seting working directory
    curdir <- getwd()
#Reading in my prefered dataset(Calling it df for short of Data frame)
  df=read.csv(paste(curdir,"/car.data",sep = ""),header=T,
              na.strings="?")
  nr <- nrow(df) # number of rows
  nc <- ncol(df) # number of columns
#delivering the data the way i want to present it
  cat ("Cars Evaluation Dataset has: ", nr," Rows", " and ",nc, " Columns")
```

```
## Cars Evaluation Dataset has:  1727  Rows  and  7  Columns
```

I then got a breif description from R before changing making any changes to the dataset

```
#Recieveing a breif description of the data set
  str(df)

## 'data.frame': 1727 obs. of  7 variables:
##  $ vhigh  : Factor w/ 4 levels "high","low","med",..: 4 4 4 4 4 4 4 4 4 4 ...
##  $ vhigh.1: Factor w/ 4 levels "high","low","med",..: 4 4 4 4 4 4 4 4 4 4 ...
##  $ X2     : Factor w/ 4 levels "2","3","4","5more": 1 1 1 1 1 1 1 1 1 1 ...
##  $ X2.1   : Factor w/ 3 levels "2","4","more": 1 1 1 1 1 1 1 1 2 2 ...
##  $ small  : Factor w/ 3 levels "big","med","small": 3 3 2 2 2 1 1 1 3 3 ...
##  $ low    : Factor w/ 3 levels "high","low","med": 3 1 2 3 1 2 3 1 2 3 ...
##  $ unacc  : Factor w/ 4 levels "acc","good","unacc",..: 3 3 3 3 3 3 3 3 3 3 ...
```

The coloumns of the dataset can be dislayed by doing the following:

```
#Retrieving the coloumn names
  names(df)

## [1] "vhigh"   "vhigh.1" "X2"      "X2.1"    "small"   "low"     "unacc"
```

The Class distribution in the dataset is the Evaluation

Table 1: Number of Instances per Class

Class	Number	Number as a Percentage
unacc	1210	70.023
acc	348	22.222
good	69	3.993
v-good	65	3.769

Below is the Maintence ratings in the dataset for the cars.

Below there is also a graph with the car Price Ratings.

Also below is a bar graph the car evaluation with their level of Acceptiblity

```
#Bar plot of the Maintence rating of the cars
  labelFreqs <- table(df$Maint)# frequency of labels
    barplot(labelFreqs,col = gray.colors(4),xlab="Car Ratings",
        main="Maintence Ratings")
#Bar Plot of the price rating of the cars
  labelFreqs <- table(df$Price)# frequency of labels
   barplot(labelFreqs,col = gray.colors(4),xlab="Car Ratings",
             main="Price Ratings")
  labelFreqs <- table(df$Evaluation)# frequency of labels
    barplot(labelFreqs,col = gray.colors(4),xlab="Acceptability",
        main="Maintence Ratings")
```

1.5 Pre-Processing

Before doing anything I began by checking my data for any missing values. Moreover I was also cheking for the amount of unique values there are for each variable. I have applied a function called sapply()which is an efficient way to pass the function as an argument and apply it to each column

```
  na_counts <- sapply(df,function(x) sum(is.na(x)))
  unique_vals<- sapply(df, function(x) length(unique(x)))
  na_counts

##   vhigh vhigh.1      X2    X2.1   small     low   unacc
##       0       0       0       0       0       0       0

  unique_vals

##   vhigh vhigh.1      X2    X2.1   small     low   unacc
##       4       4       4       3       3       3       4
```

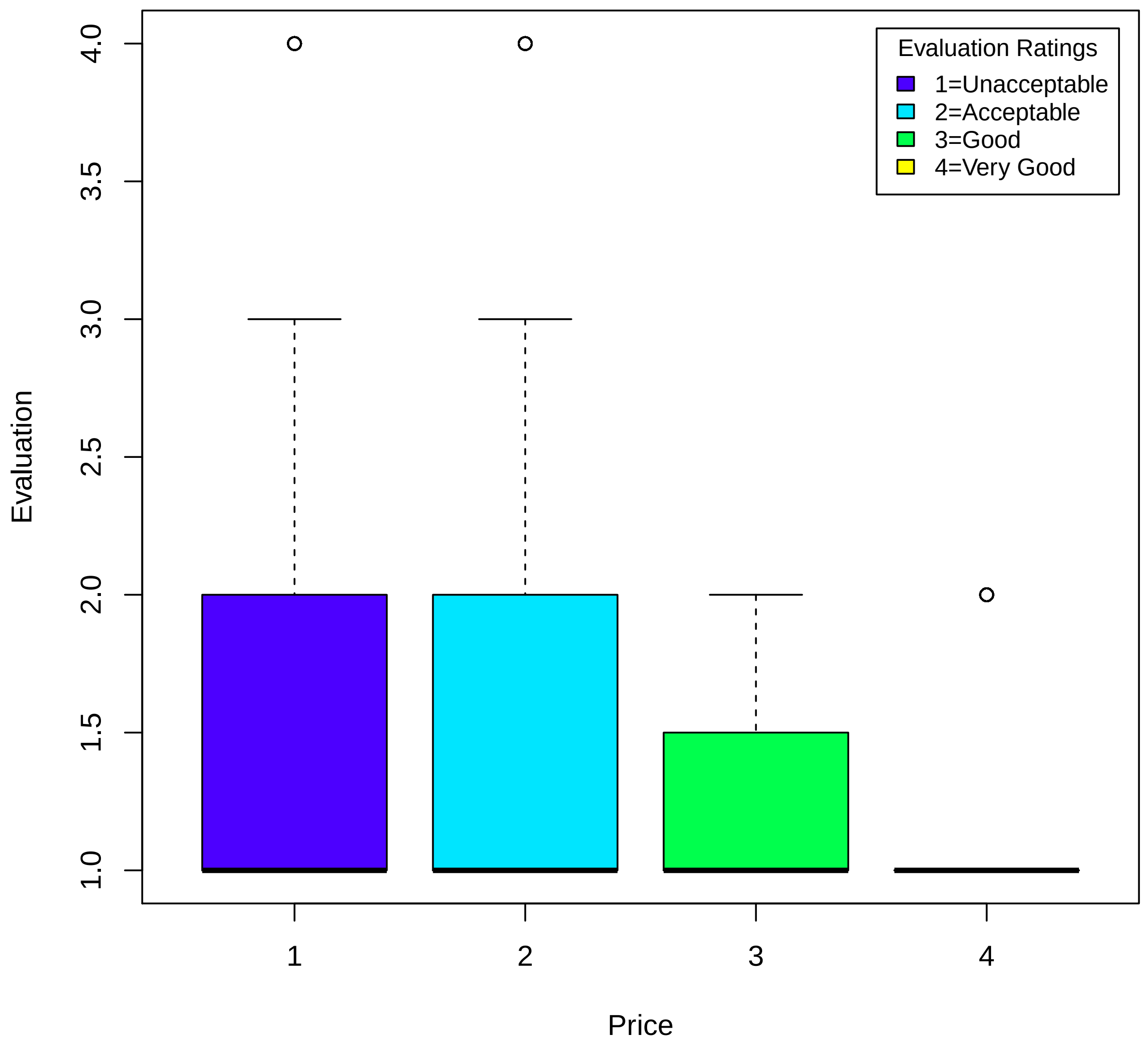
Now I going to begin applying changes that i feel need to be applied in order for accuring a prediction from the dataset. I am going to begin with creating another dataframe so i have a back up.

```
#Creating a back up data frame called nd(New Data)
 nd <- df

#Re-definng the coloumn names to something more understandable
colnames(nd)[which(names(nd) == "vhigh")] <- "Price" #This is the price of the car
colnames(nd)[which(names(nd) == "vhigh.1")] <- "Maint"#This is the Maintence of the car
colnames(nd)[which(names(nd) == "X2")] <- "Doors"#This is the number of doors the car has
colnames(nd)[which(names(nd) == "X2.1")] <- "Persons"#The number of the people the car can hold
colnames(nd)[which(names(nd) == "small")] <- "Boot"#This is the boot size
colnames(nd)[which(names(nd) == "low")] <- "Safety"#This is the safety rating of the car
colnames(nd)[which(names(nd) == "unacc")] <- "Evaluation"#This is the evaluation of the car

#Changing the values to numbers so that it is easier to do predictions
#One By one all features are going to be changed
nd$Price<-as.character(nd$Price)
nd$Price[nd$Price=="vhigh"] <- "4"
nd$Price[nd$Price=="high"] <- "3"
nd$Price[nd$Price=="med"] <- "2"
nd$Price[nd$Price=="low"] <- "1"
nd$Price = as.numeric(nd$Price)

nd$Maint<-as.character(nd$Maint)
nd$Maint[nd$Maint=="vhigh"] <- "4"
nd$Maint[nd$Maint=="high"] <- "3"
nd$Maint[nd$Maint=="med"] <- "2"
nd$Maint[nd$Maint=="low"] <- "1"
nd$Maint = as.numeric(nd$Maint)

nd$Doors<-as.character(nd$Doors)
nd$Doors[nd$Doors=="5more"] <- "5"
nd$Doors[nd$Doors=="4"] <- "4"
nd$Doors[nd$Doors=="3"] <- "3"
nd$Doors[nd$Doors=="2"] <- "2"
nd$Doors[nd$Doors=="1"] <- "1"
```

```
nd$Doors = as.numeric(nd$Doors)

nd$Persons<-as.character(nd$Persons)
nd$Persons[nd$Persons=="more"] <- "5"
nd$Persons[nd$Persons=="4"] <- "4"
nd$Persons[nd$Persons=="2"] <- "2"
nd$Persons = as.numeric(nd$Persons)

nd$Boot<-as.character(nd$Boot)
nd$Boot[nd$Boot=="small"] <- "1"
nd$Boot[nd$Boot=="med"] <- "2"
nd$Boot[nd$Boot=="big"] <- "3"
nd$Boot = as.numeric(nd$Boot)

nd$Safety<-as.character(nd$Safety)
nd$Safety[nd$Safety=="low"] <- "1"
nd$Safety[nd$Safety=="med"] <- "2"
nd$Safety[nd$Safety=="high"] <- "3"
nd$Safety = as.numeric(nd$Safety)

nd$Evaluation<-as.character(nd$Evaluation)
nd$Evaluation[nd$Evaluation=="unacc"] <- "1"
nd$Evaluation[nd$Evaluation=="acc"] <- "2"
nd$Evaluation[nd$Evaluation=="good"] <- "3"
nd$Evaluation[nd$Evaluation=="vgood"] <- "4"
nd$Evaluation = as.numeric(nd$Evaluation)
```

Box plot for the Evaluation against the price

```
attach(nd)
boxplot(Evaluation ~ Price,
        xlab="Price",ylab="Evaluation",
        col=topo.colors(4))
legend("topright", inset=.02, title="Evaluation Ratings",
   c("1=Unacceptable","2=Acceptable","3=Good","4=Very Good"), fill=topo.colors(4), horiz=FALSE, cex=0.8
```



Price; 1 = Low 2 = Medium 3 = High 4 = Very High

2 Modelling/ Classification

The approach taken to modelling the data set was the random forest technique this approach was taken because it outputs a prediction matrix's of arbitrary size in this case there are 4 factors which are being predicted which result in a prediction matrix size 16.Random forest combines flexibility and power into a learning method as the entity uses only a small random percentage of the full dataset. Moreover, the technique used can handle extremely large datasets which might cause other models to fail. Furthermore random forest is appropriate for missing data as well as categorical or continous details.

Dividing the dataset into training and testing subset

The system sets the class label as a factor after being transformed to a Numeric Value

```
#Setting the Label as a Factor avoid to regression to face matheatical errors
nd$Evaluation <- as.factor(nd$Evaluation)
```

The system begins with spliting the data into training and testing sets.

```
#Importing the Correct Library
library(caret)

## Loading required package:  lattice
## Loading required package:  ggplot2

#splitting the data set in to 2 parts one for training and the other for testing
#80 percent set is the traing
inTrain <- createDataPartition(y=nd$Evaluation,
                               p=.8,list=FALSE)
#defining the training set as the variable created above
training <- nd[inTrain,]
#defining the testing data set from what is left over from the variable created above
testing <- nd [-inTrain,]
#Performing a test to check the the total number of records both train/test are equal to the orgininal 
#This should result in a display of true
nrow(training)+nrow(testing)==nrow(nd)

## [1] TRUE
```

The program creates a fit model on which the classification can begin

```
#create the variable on which the evaluation is being predicted against all others
fitModel <- train(Evaluation ~ .,data=training,
                  method="rf",prox=TRUE)

## Loading required package:  randomForest
## randomForest 4.6-12
## Type rfNews() to see new features/changes/bug fixes.
##
## Attaching package:  'randomForest'
## The following object is masked from 'package:ggplot2':
##
##    margin

#Displaying the final model before the predictions
fitModel$finalModel

##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry, proximity = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 6
##
##         OOB estimate of  error rate: 1.52%
## Confusion matrix:
##     1   2  3  4 class.error
## 1 961   4  3  0 0.007231405
## 2   2 299  5  2 0.029220779
## 3   1   1 53  1 0.053571429
## 4   0   1  1 50 0.038461538
```

```
df_C <- classCenter(training[,c(3,4)],
                    training$Evaluation,fitModel$finalModel$prox)
df_C <- as.data.frame(df_C)
df_C$Evaluation <- rownames(df_C)
```

Displaying the fitmodel

```
#Getting the system to display the fit model
summary(fitModel)

##                  Length  Class      Mode
## call                   5 -none-     call
## type                   1 -none-     character
## predicted           1384 factor     numeric
## err.rate            2500 -none-     numeric
## confusion             20 -none-     numeric
## votes               5536 matrix     numeric
## oob.times           1384 -none-     numeric
## classes                4 -none-     character
## importance             6 -none-     numeric
## importanceSD           0 -none-     NULL
## localImportance        0 -none-     NULL
## proximity        1915456 -none-     numeric
## ntree                  1 -none-     numeric
## mtry                   1 -none-     numeric
## forest                14 -none-     list
## y                   1384 factor     numeric
## test                   0 -none-     NULL
## inbag                  0 -none-     NULL
## xNames                 6 -none-     character
## problemType            1 -none-     character
## tuneValue              1 data.frame list
## obsLevels              4 -none-     character
## param                  1 -none-     list
```

Building the prediction model

```
#Test the model for this run
pred <- predict(fitModel, testing)
testing$correctPred <- pred==testing$Evaluation
```

Retriving the Result

```
#Assigning and displaying the results
results <- table(pred,testing$Evaluation)
results

##
## pred   1   2   3   4
##    1 237   2   0   0
##    2   4  73   0   1
##    3   0   1  13   0
##    4   0   0   0  12
```

Testing the values to see what the Accuracy is

```
#Assinging the accuracy and displaying it
accuracy <- sum(diag(results)) / sum(results) * 100
accuracy

## [1] 97.66764
```

3 Improving Performance