University of Oklahoma

ECE 5970-980

Group Assignment

Exploratory Data Analysis on carData

Group Members:

Nabil Asfari

Mohamat Eirban Ali Bin Kaja Najumudeen

Dataset and pre-processing

For this assignment, the dataset was already given to us by Dr. MacDonald. The dataset's title is carData; the dataset describes few data component in car production. The dataset contains 7 columns and 1728 rows of data; the 7 columns describe the price, maintenance cost, number of doors (car type), trunk size, number of people, safety measure and ranking.

Firstly, we already made sure there are no unfilled space or 'NA' in any of the dataspace. We also have noticed that the word 'more' was used to describe in some instance the data of 'Persons' and in some case, the number of door is described with '5more'. These cases could be a mistype or filling in error of which we could not determine with the limited information we have. Also, we know the data type in this data set is all categorical as such the use of word 'more' could just be considered as another category to be considered; taking these as an assumption, firstly we have done a bar plot for all the data columns to find the frequency count for each category.

Frequency Count for each class in data set:

Sample code for bar plot frequency count:

```
> library(data.table)
data.table 1.11.8  Latest news: r-datatable.com
> library(ggplot2)
> datax<-data.frame(read.csv(file="carData.csv"))
> car_price  = table(datax$Price)
> Price_viz = barplot(car_price, main="Price of car  ", ylab= "Count")
```

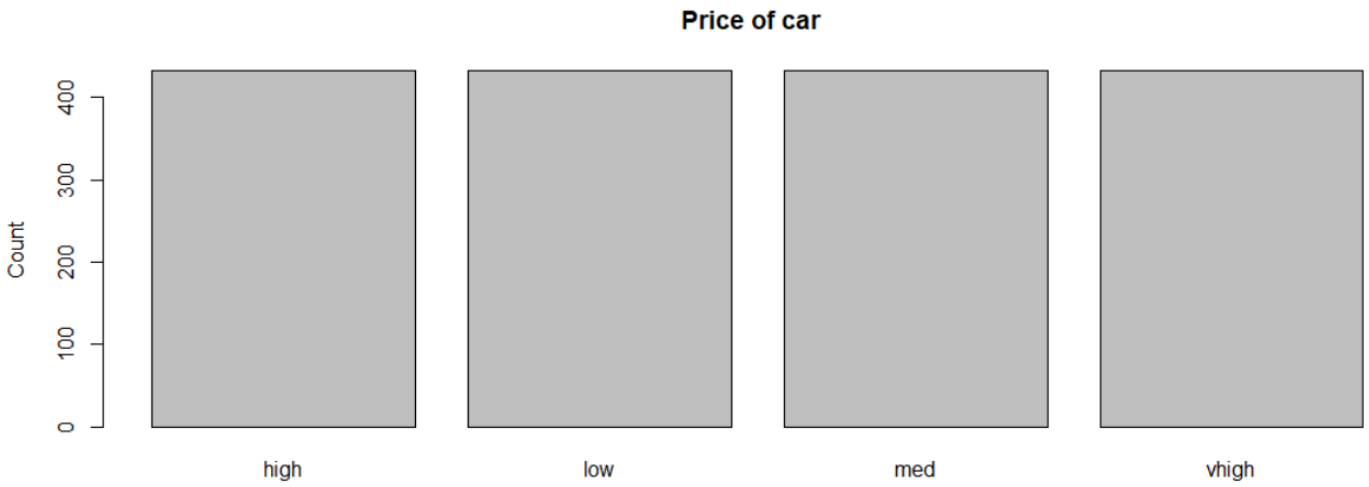

Figure 1. Price of the car

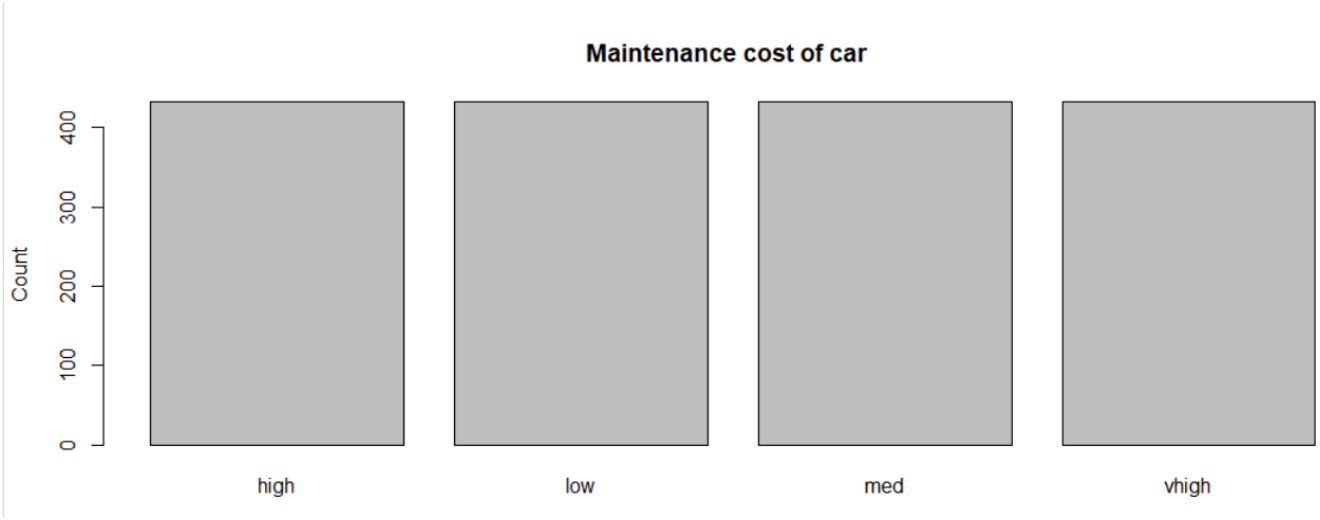

Figure 2. Maintenance of the car

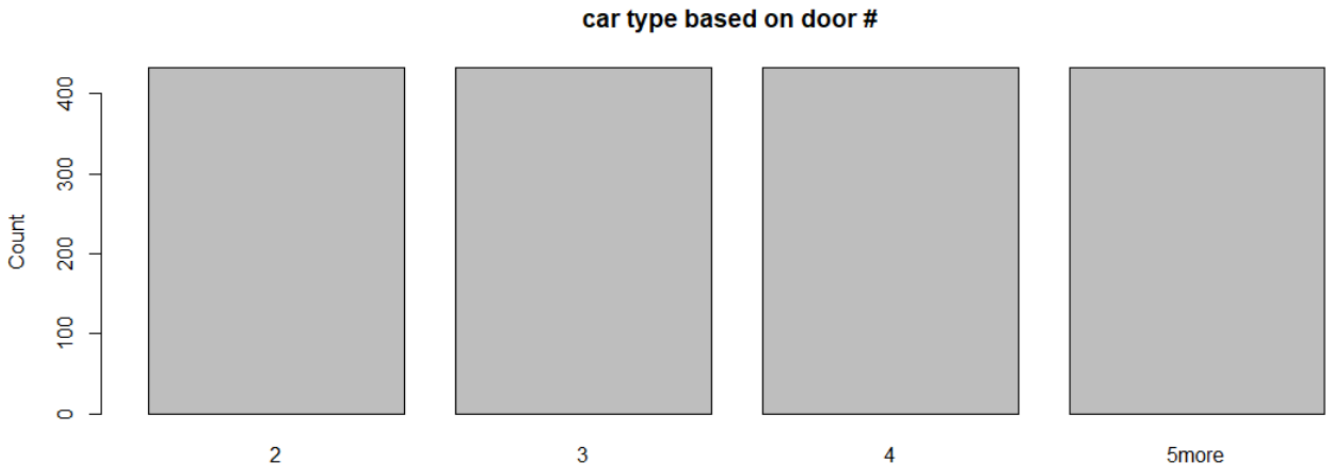

Figure 3. Car type based on door #

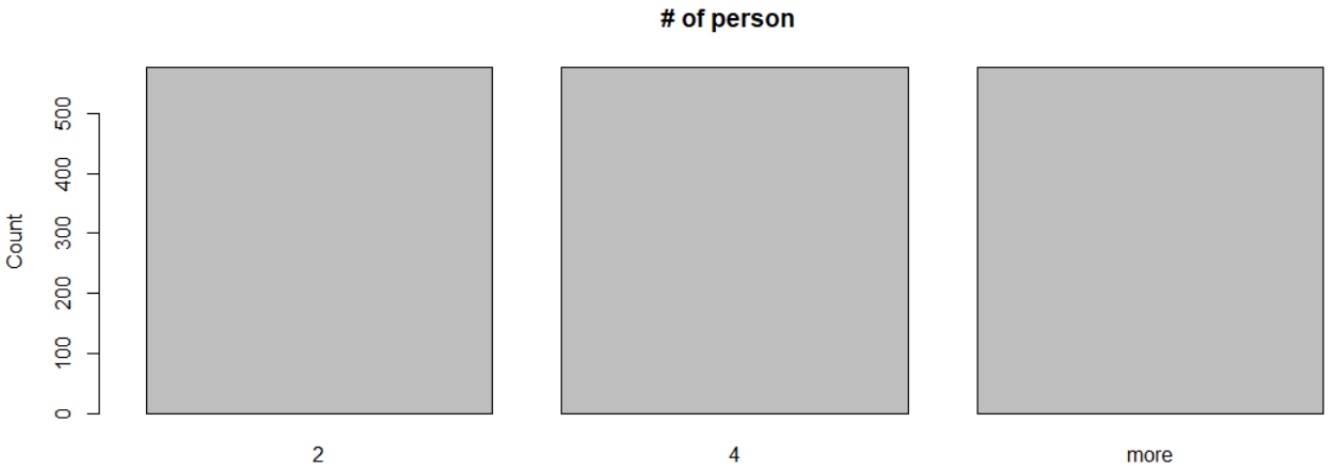

Figure 4. # of person

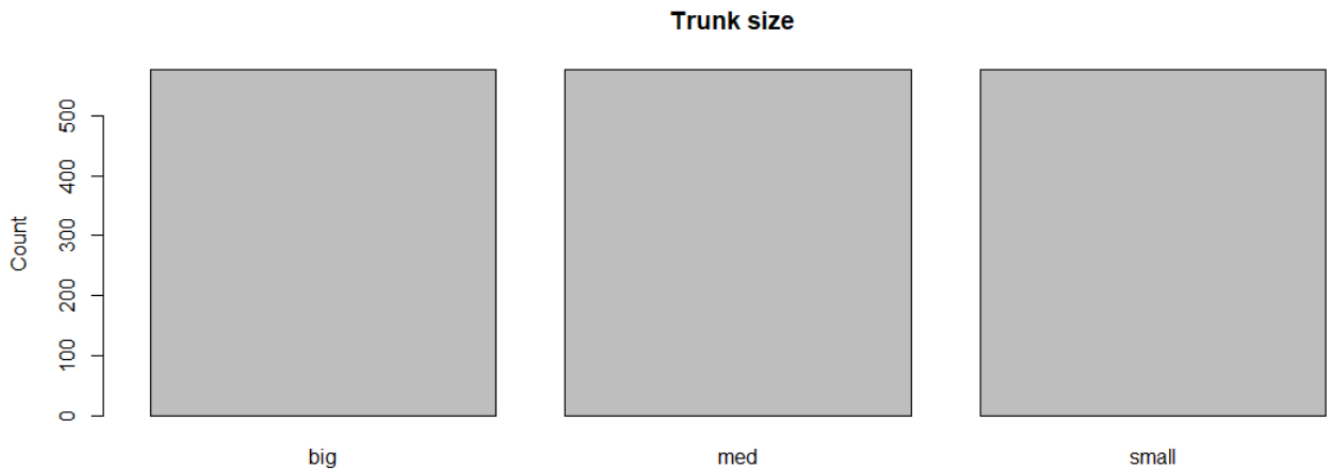

Figure 5. Trunk size

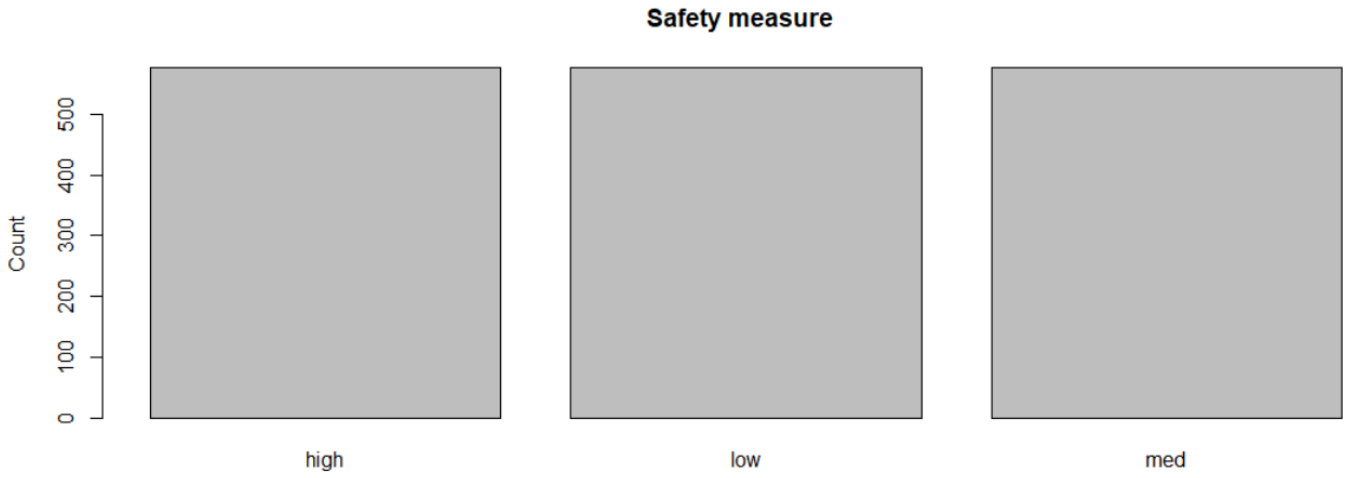

Figure 6. Safety measure

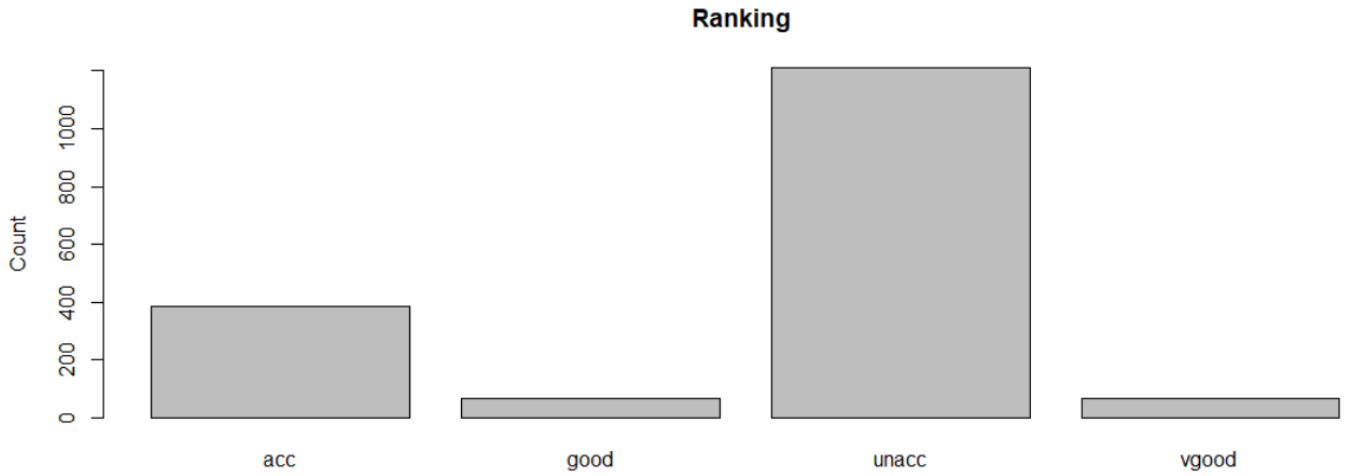

Figure 7. Ranking

Figure 1 to Figure 7 shows us the frequency count for each class of data based on the given category. We noticed that for all the variables, the frequency count was equally distributed amongst the categories except for the ‘Ranking’ variables. Based on this, we can see that the ‘Ranking’ variable was meant to be the resultant variable of all the other variable which makes ‘Ranking’ variable to be product outcome of the results in the other variables.

Correlation plot between variables and our target 'Ranking':

Firstly, for the demonstration purpose we showed there are two or even more ways to do correlation for data such as this. In this case, we created a simple plot from coding in R studio. We also did the exact same thing using the Rattle package in R.

Sample code for correlation bar plot between variables and Ranking:

```
> dat <- data.frame(table(datax$Price,datax$Ranking))
> names(dat) <- c("Price","Ranking","Count")
>
> ggplot(data=dat, aes(x=Price, y=Count, fill=Ranking)) + geom_bar(stat="identity")
```

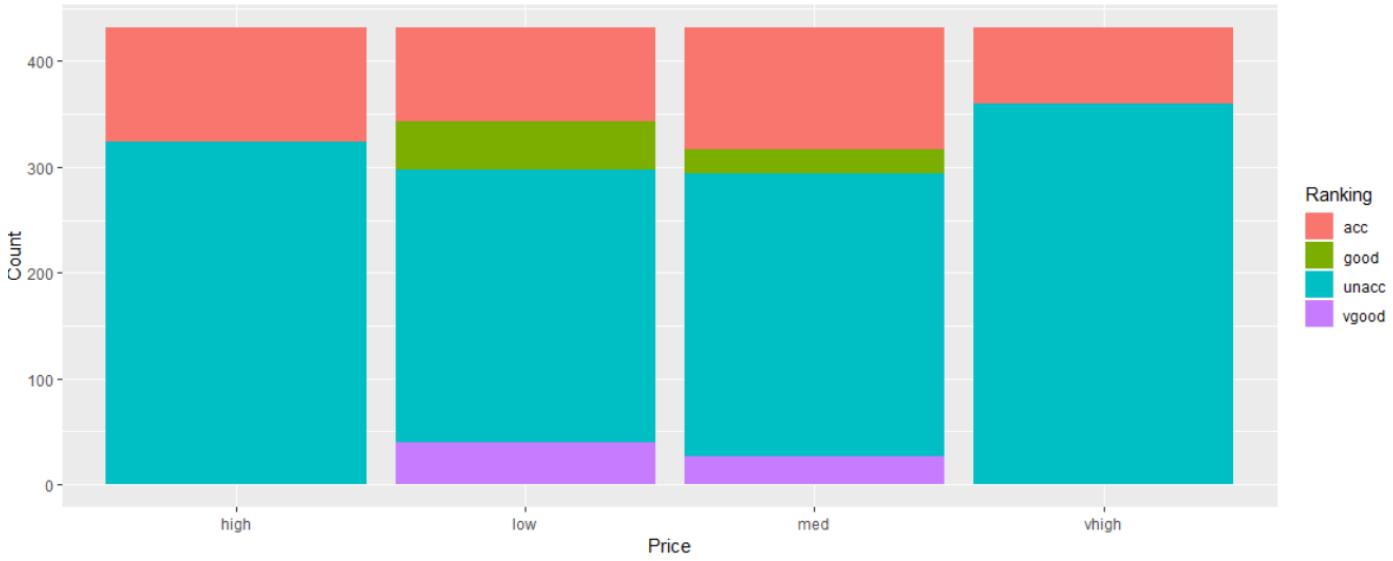

Figure 8. Price plot in R studio

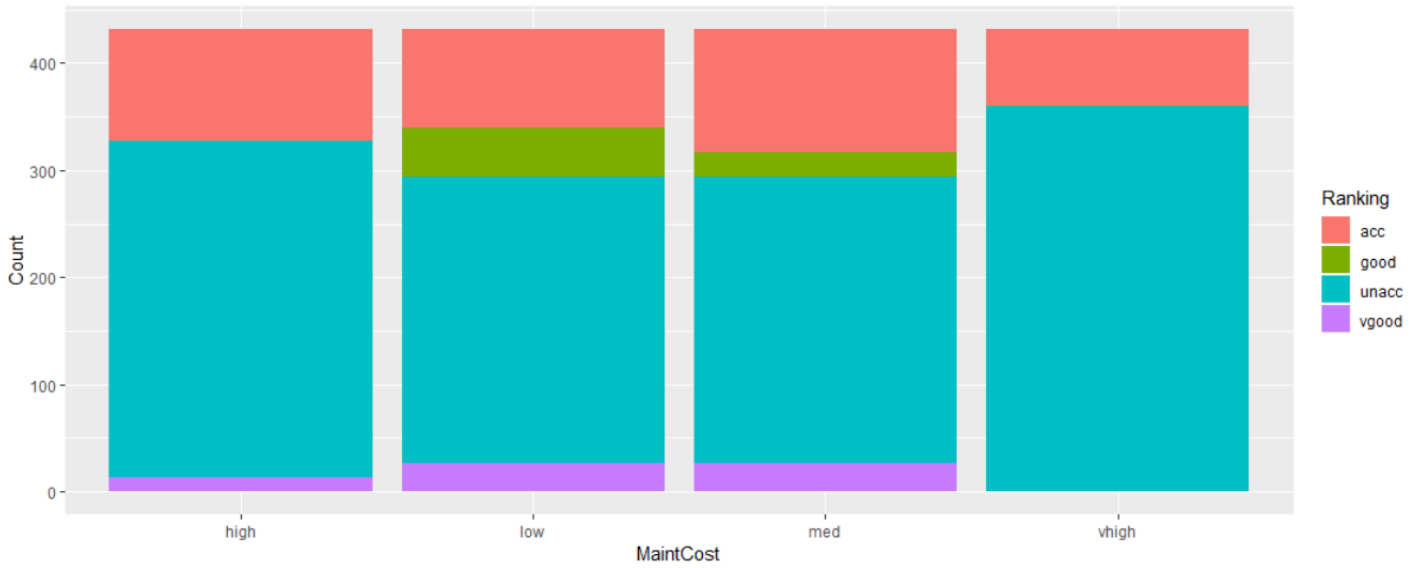

Figure 9. Maintenance cost plot in R studio

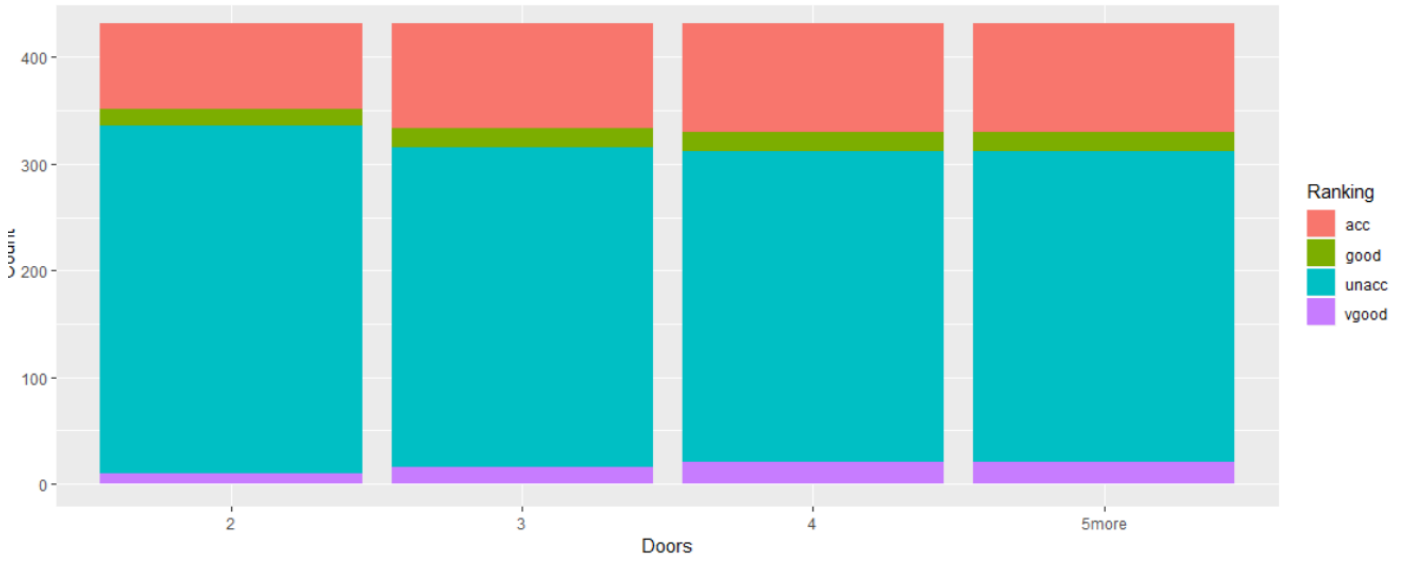

Figure 10. # of Doors plot in R studio

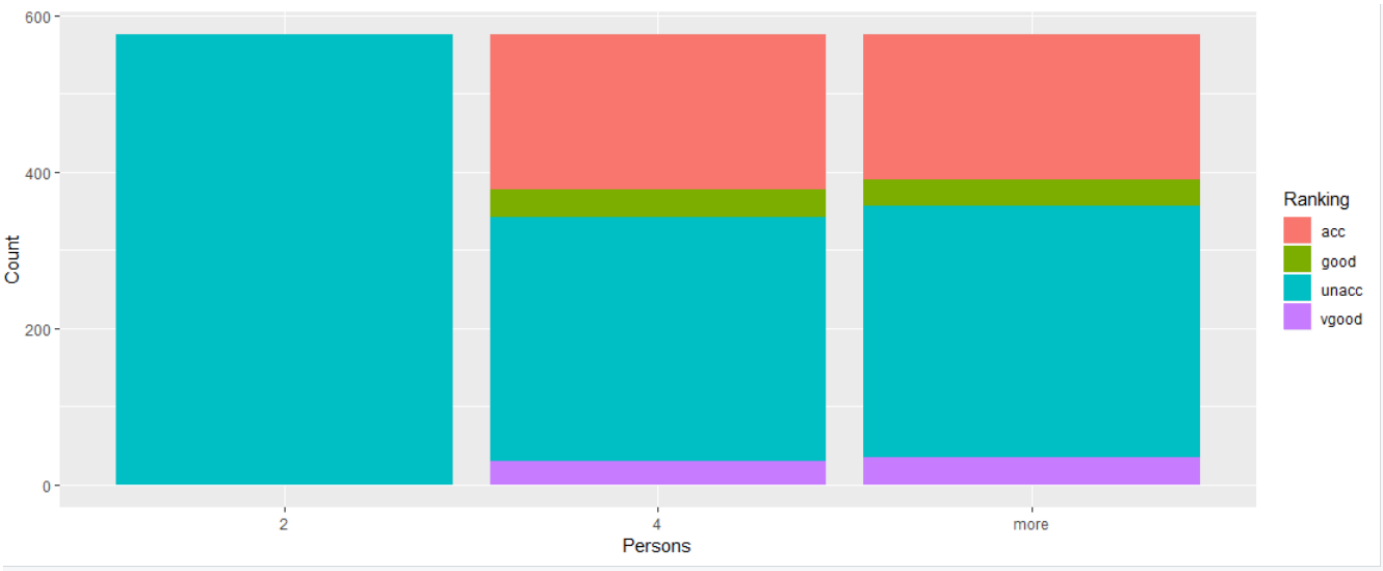

Figure 11. # of Person plot in R studio

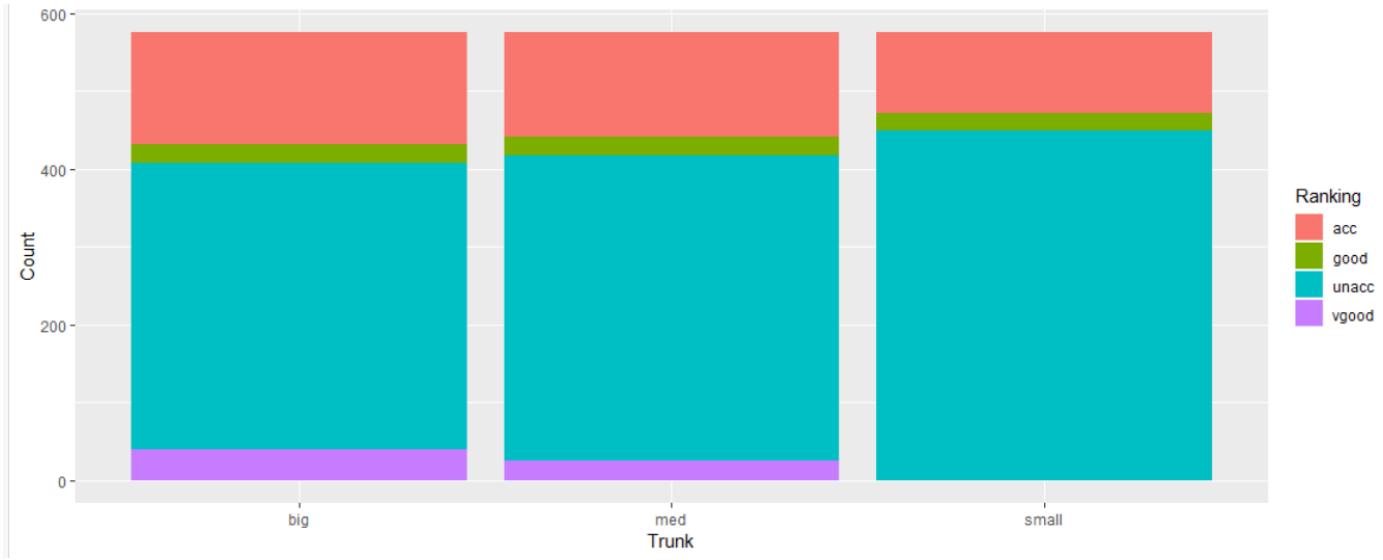

Figure 12. Trunk size plot in R studio

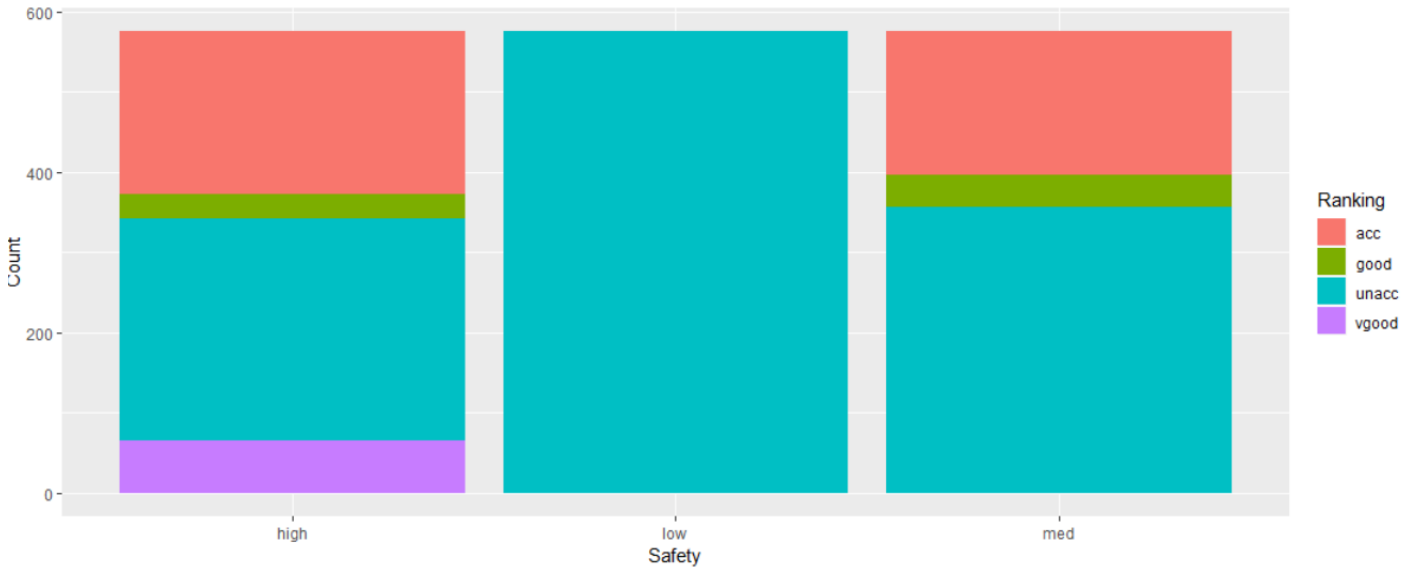

Figure 13. Safety plot in R studio

We used each variables distribution in their category and compare it to how they correlate to the frequency of each Ranking that occurs. As such, just by looking at Figure 8 to Figure 13, we can sort of see that almost all the variables have somewhat positive correlation to the Ranking variable except for the variable ‘Price’.

Correlation plot using Rattle's Explore function:

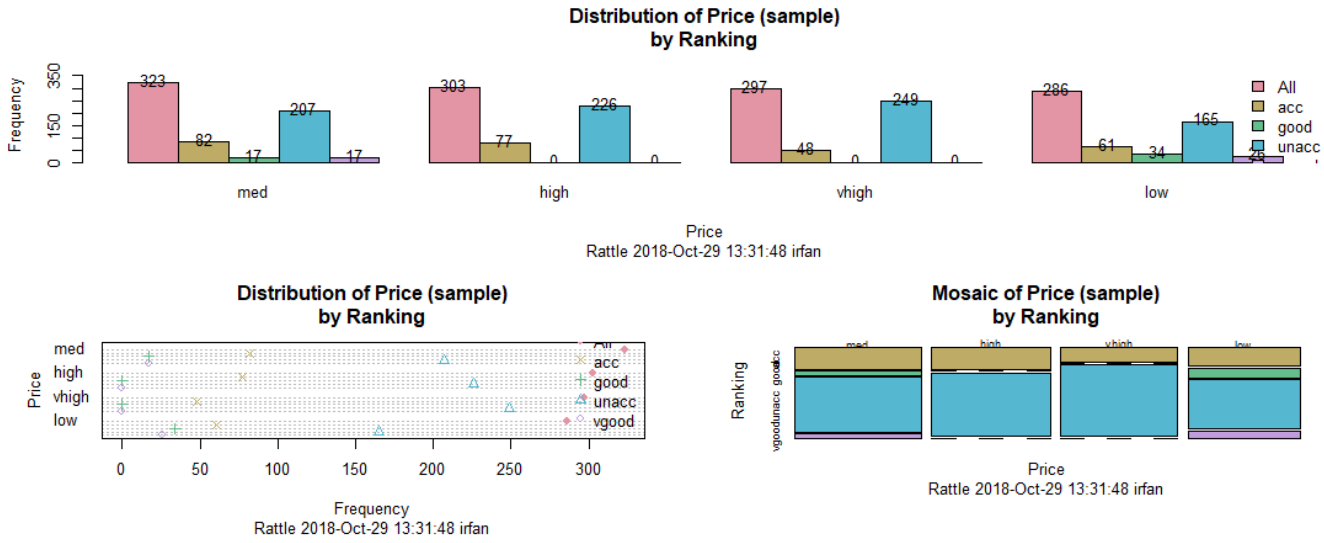

Figure 14. Correlation plot using Explore function in Rattle

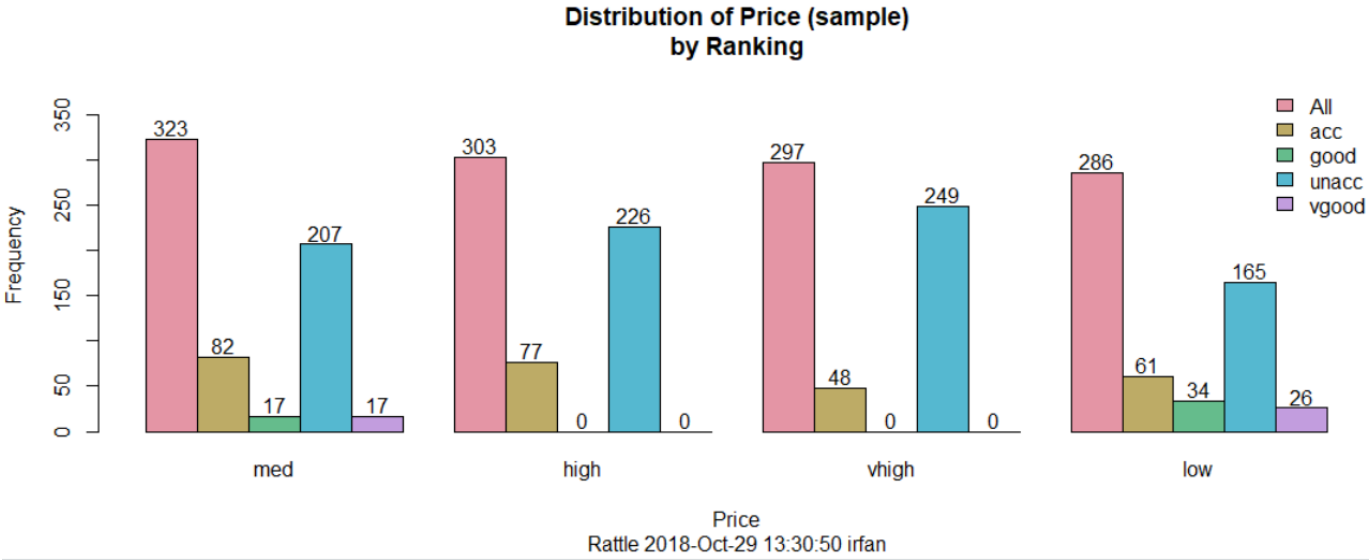

Figure 15. Distribution of Price by Ranking plot in Rattle

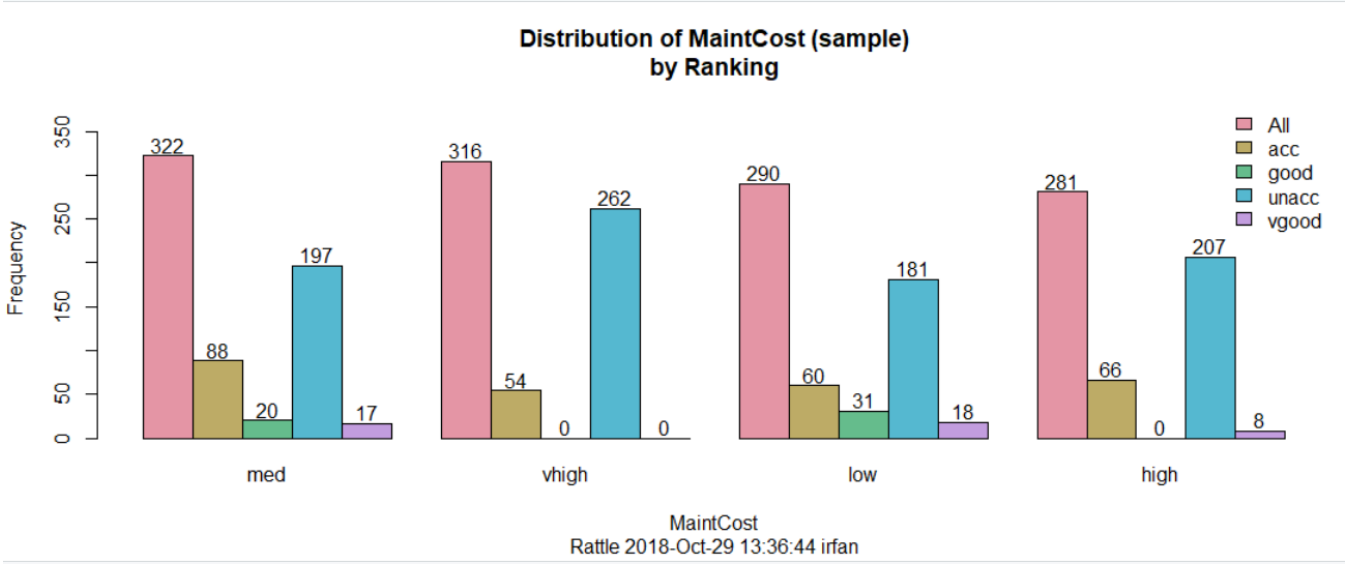

Figure 16. Distribution of Maintenance Cost by Ranking plot in Rattle

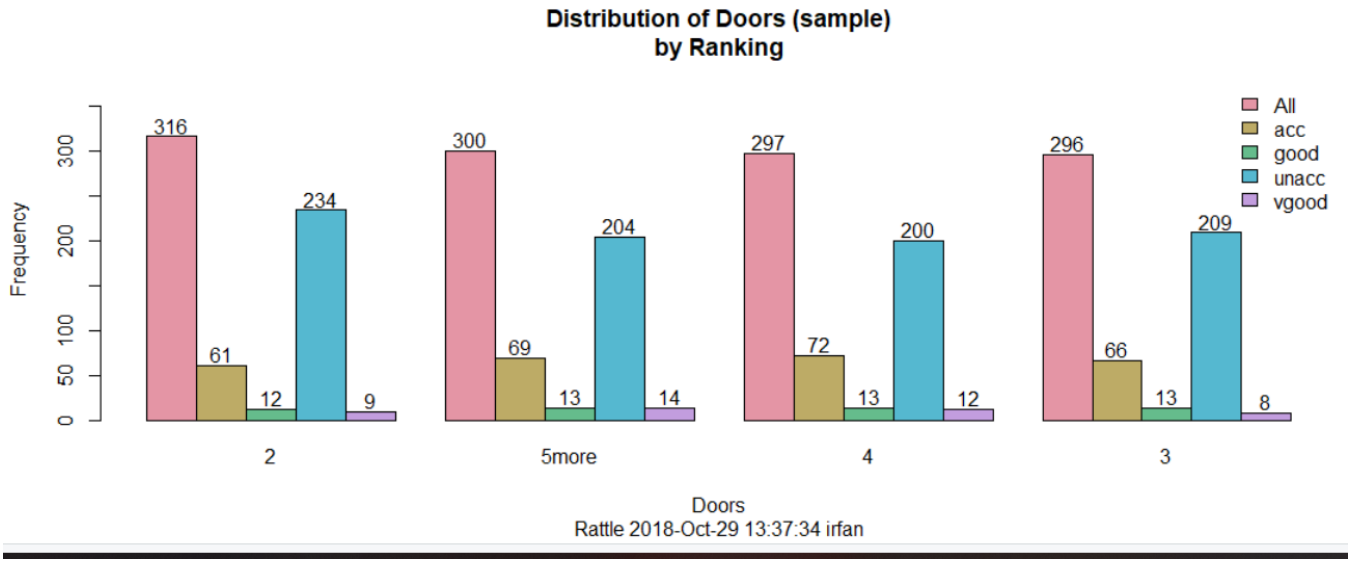

Figure 17. Distribution of Doors by Ranking plot in Rattle

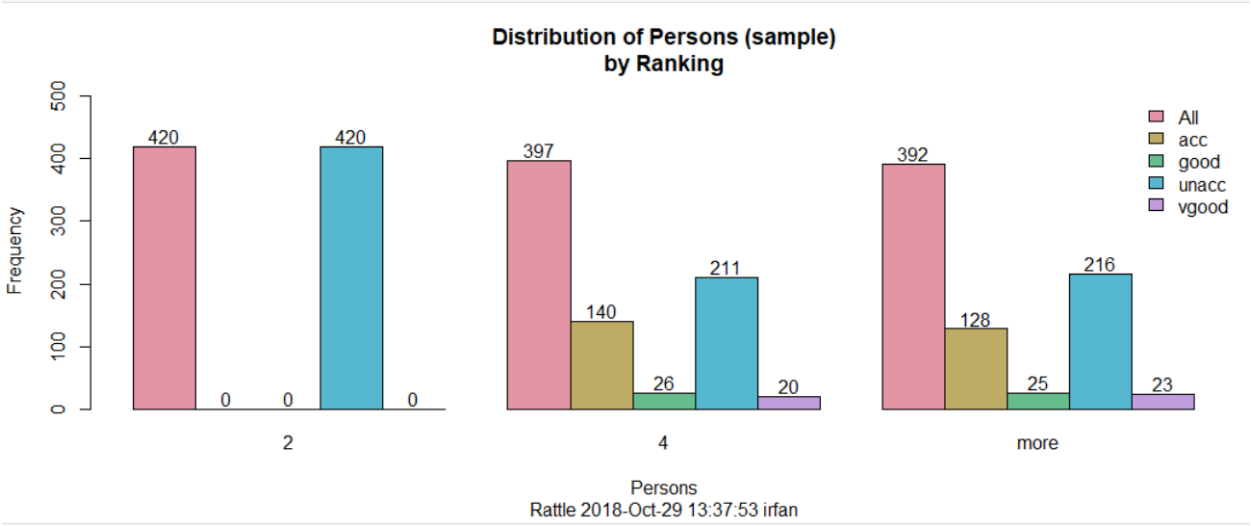

Figure 18. Distribution of Persons by Ranking plot in Rattle

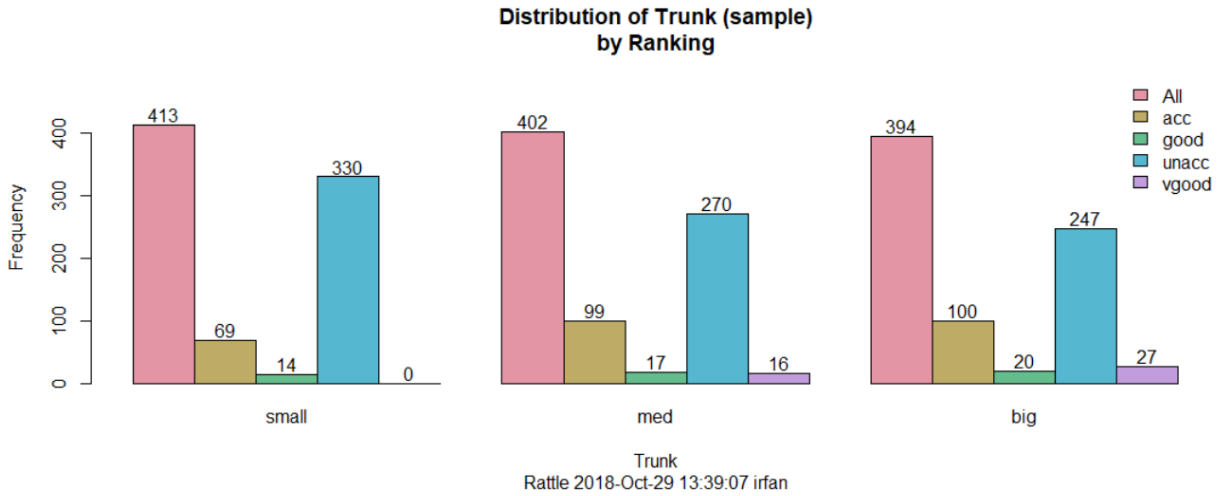

Figure 19. Distribution of Trunk by Ranking plot in Rattle

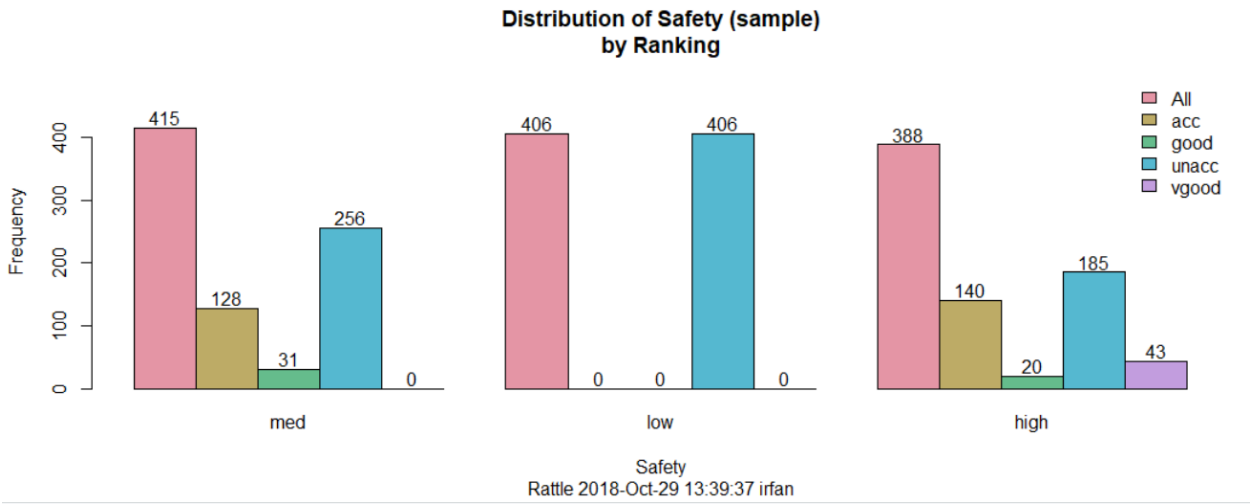

Figure 20. Distribution of Safety by Ranking plot in Rattle

Figure 14 to Figure 20 just shows the same outcome as Figure 8 to Figure 13 except that these plots were obtained using the ‘Explore’ tab in Rattle package for the R studio.

Decision Tree Modelling using rattle in R:

Decision Tree Chart:

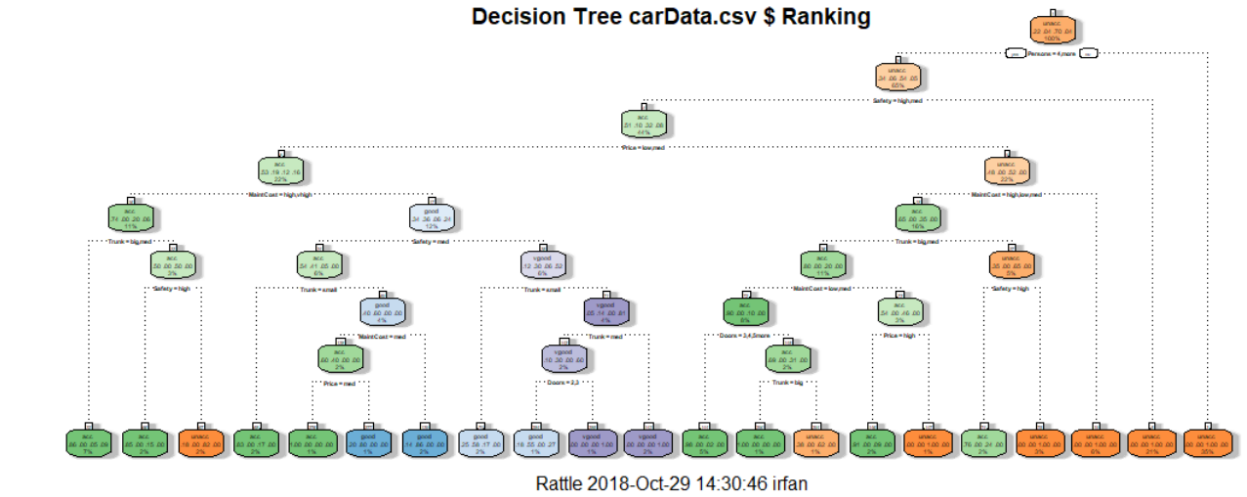

Figure 21. Decision Tree Chart

The parameters used to build the decision tree:

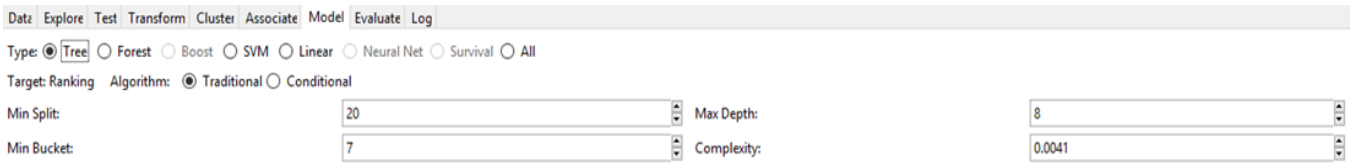

Figure 22. Parameters used for Decision Tree Chart

The parameters were decided by multiple trial and error as to get a lower overall error in the confusion matrix generated from the 'Evaluate' tab in Rattle. We also chose the complexity value based on lower xerror value when we initially ran the model with a complexity of 0.001; the summary generated for that model shows us the xerror will be lower at complexity value of 0.0041.

```
Error matrix for the Decision Tree model on carData.csv [validate] (counts):

       Predicted
Actual  acc good unacc vgood Error
  acc    49    3     4     0  12.5
  good    0    7     0     0   0.0
  unacc   7    0   178     0   3.8
  vgood   2    1     0     8  27.3

Error matrix for the Decision Tree model on carData.csv [validate] (proportions):

       Predicted
Actual   acc good unacc vgood Error
  acc   18.9  1.2   1.5   0.0  12.5
  good   0.0  2.7   0.0   0.0   0.0
  unacc  2.7  0.0  68.7   0.0   3.8
  vgood  0.8  0.4   0.0   3.1  27.3

Overall error: 6.6%, Averaged class error: 10.9%

Rattle timestamp: 2018-10-29 14:32:47 irfan
=====================================================================
```

Table 1. Error Matrix for Decision Tree Model #1

```
Error matrix for the Decision Tree model on carData.csv [test] (counts):

       Predicted
Actual  acc good unacc vgood Error
  acc    56    2     2     0   6.7
  good    0   11     0     0   0.0
  unacc   9    0   169     0   5.1
  vgood   3    0     0     8  27.3

Error matrix for the Decision Tree model on carData.csv [test] (proportions):

       Predicted
Actual   acc good unacc vgood Error
  acc   21.5  0.8   0.8   0.0   6.7
  good   0.0  4.2   0.0   0.0   0.0
  unacc  3.5  0.0  65.0   0.0   5.1
  vgood  1.2  0.0   0.0   3.1  27.3

Overall error: 6.2%, Averaged class error: 9.775%

Rattle timestamp: 2018-10-29 14:33:10 irfan
=====================================================================
```

Table 2. Error Matrix for Decision Tree Model #2

From Table 1. And Table 2. of the confusion matrix, we can observe that the overall error for the validation data comes to 6.6% and the overall error for test data comes to 6.2%. For both

data sets, the main contributor to the error comes from the prediction of the variable ‘Ranking’ very good.

SVM modelling to find optimal svm model of the car data:

For the single vector machine analysis, we used the provided code to train our dataset and produce confusion matrix for the test data set as to observe whether we can obtain an optimal SVM model of our carData. To simplify the optimization process, we were able to discover the tune function that we can use to identify the optimal gamma and epsilon value for our SVM model. Below in the original code, we have included a small modification that adds the tune function and from that, we used the obtained gamma and epsilon value for our SVM model.

```
tuned_parameters <- tune.svm(Ranking~., data = training, gamma = 10^(-5:-1), cost = 10^(-3:1))

print(tuned_parameters)
```

```
> tuned_parameters <- tune.svm(Ranking~., data = training, gamma = 10^(-5:-1), cost = 10^(-3:1))
> print(tuned_parameters)

Parameter tuning of ‘svm’:

- sampling method: 10-fold cross validation

- best parameters:
 gamma cost
   0.1   10

- best performance: 0.03666667
```

Figure 23. Modified code to add the tune function

```
#The best model, use the value of gamma and cost found from tuned_parameters print function

my_model <- svm(Ranking ~., data = training,  kernel = "radial", gamma = 0.1, cost = 10)

# Pass the training (YIKES!) data set through the model

results<-predict(my_model,training)

# Look at the confusion matrix

table(training$Ranking,results)

# Pass the test data set through the model
```

```
results<-predict(my_model,test)

# Look at the confusion matrix

table(test$Ranking,results)
```

```
> # Look at the confusion matrix
> table(training$Ranking,results)
       results
        acc good unacc vgood
  acc   258    3     0     2
  good    0   46     0     5
  unacc   6    2   833     0
  vgood   0    0     0    45
> # Pass the test data set through the model
> results<-predict(my_model,test)
> # Look at the confusion matrix
> table(test$Ranking,results)
       results
        acc good unacc vgood
  acc   116    3     1     1
  good    0   14     0     4
  unacc   6    0   363     0
  vgood   2    0     0    18
>
```

Table 3. Confusion Matrix Table

Table 3 is the confusion matrix of both the training data and test data being ran into the optimal SVM model we built. As you can observe, the main culprit for the error in the decision tree model was the category very good but in this SVM model, it produced a lower error for the test data at about only 11 percent compared to the 27.3 percent error from the decision tree model.

Self-Organizing Map (SOM) for the carData

We ran the given SOM code in our R studio to obtain the needed results:

Plot some stuff

plot(som_model,type="dist.neighbours",palette.name=shadesOfGrey)

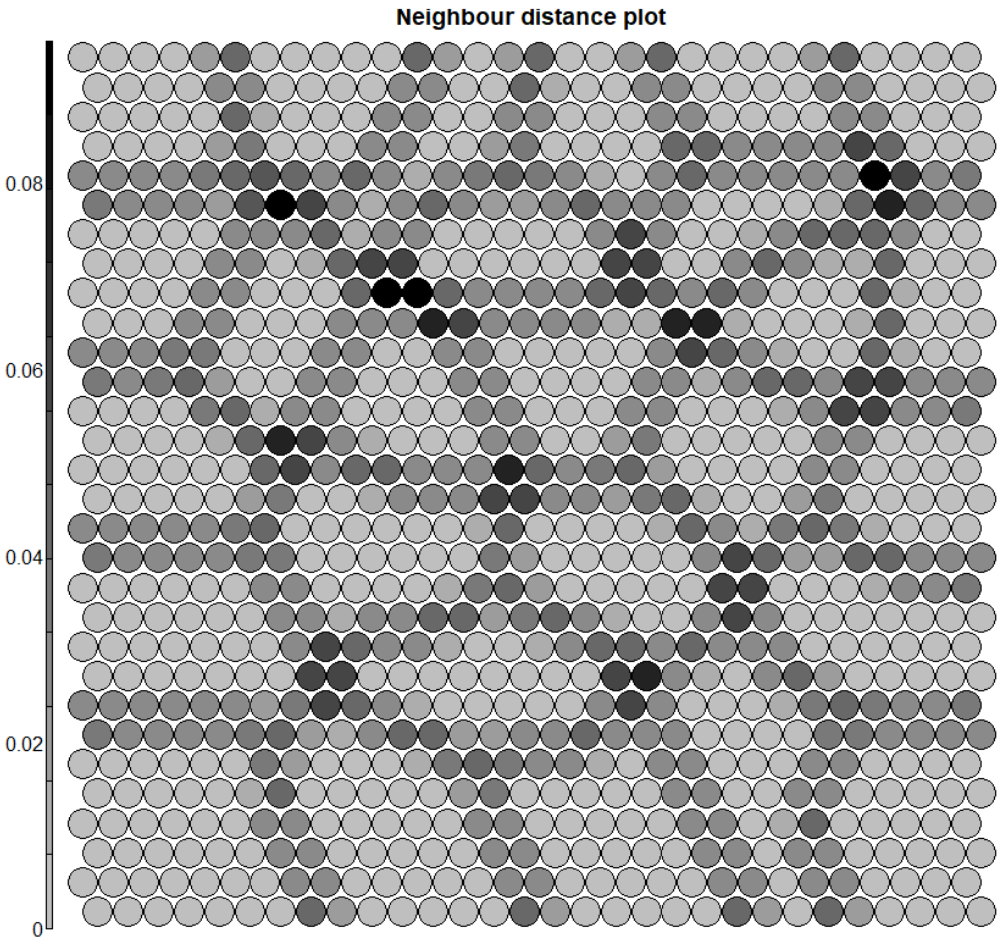

Figure 24. Self-Organizing Map (SOM) for the carData

Figure 24. is the plot of neighbor distance we obtained for the SOM model of carData set. As you know, this plot is also known as U-matrix which is a visualization of the distance between the nodes and its neighbors. As you can see from the plot, the nodes with low distance indicates high similarity while the nodes with large distance indicates dissimilarity and we may even assume that those high distance nodes form boundaries that can indicate clusters in our data.

Now use HAC to estimate the number of clusters

d<-dist(som_model$codes[[1]])

cah<-hclust(d,method="ward.D")

plot(cah,hang=-1)

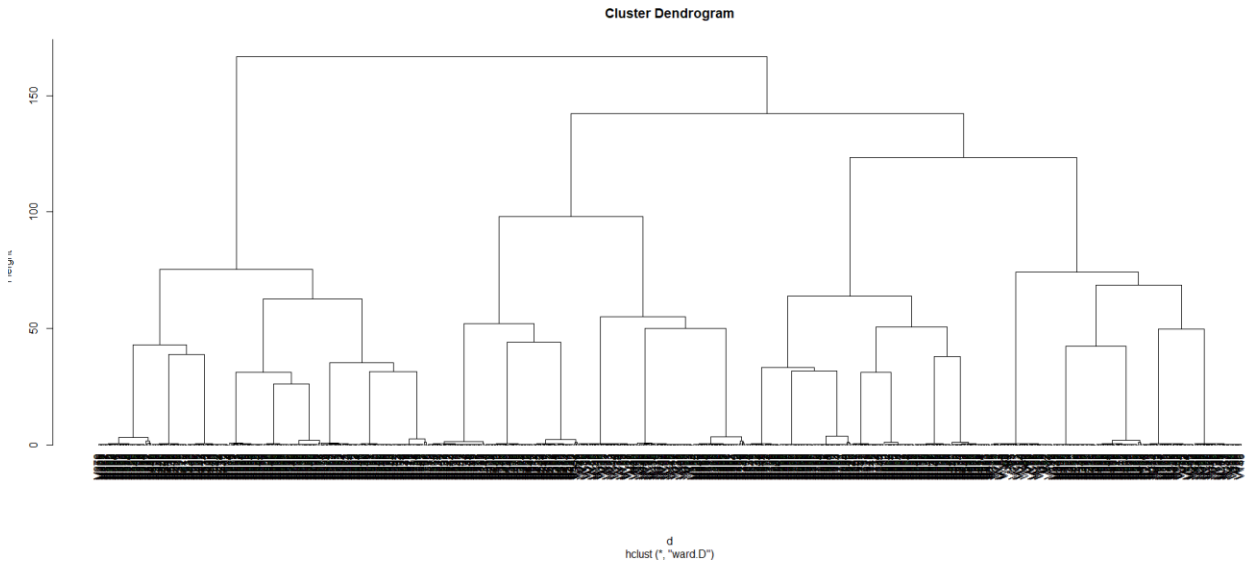

Figure 25. Cluster Dendrogram

The Figure 25. shows the clustering we obtained from applying the hierarchal clustering onto the SOM model we previously built.

Show the clusters in the dendrogram

numClust=10

groupes<-cutree(cah,k=numClust)

plot(som_model,type="mapping",bgcol=c("yellowgreen","steelblue1","sienna1",

"red","green","yellow","blue",

"steelblue2","orange","purple","steelblue3")[groupes])

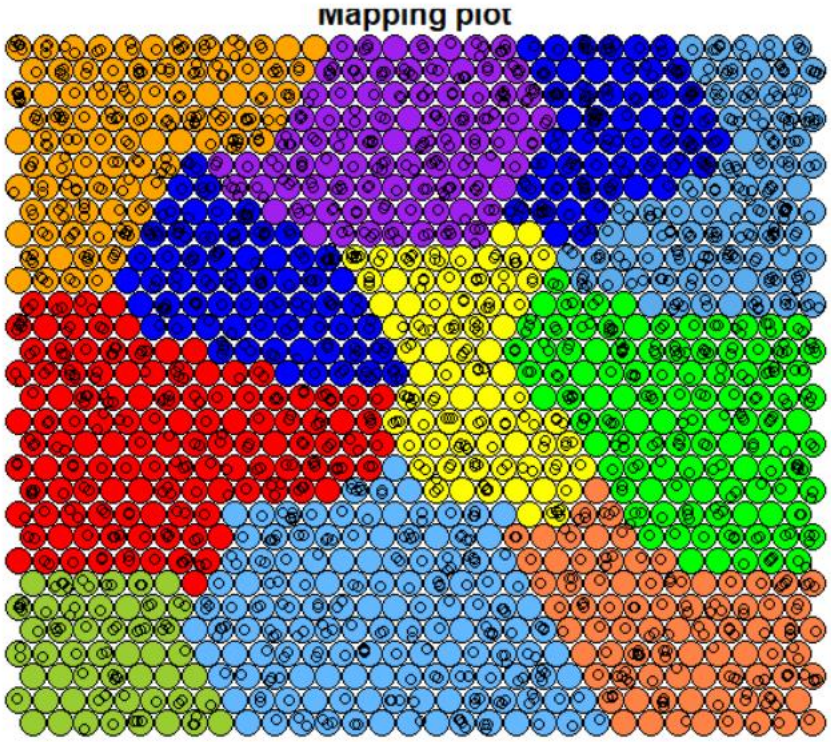

Figure 26. Mapping Plot

Using the hierarchal clustering we did previously, we can sort of provide visualization to the SOM plot by deciding on the number of clusters we want to obtain; based on the number chosen for the cluster, the dendrogram will be cut at the appropriate height as to provide the clusters.

```
# Now let’s examine where the VGOOD rankings are

ratevgood<-NULL

for (i in 1:(xdim*ydim)){

  matches<-which(som_model$unit.classif==i)

  cellTotal<-length(matches)

  keepthese<-data[matches,]

  matches<-which(keepthese$Ranking=='vgood')
```

```
  vgoodTotal<-length(matches)
  ratevgood[i]<-0
  if (cellTotal>0){
    ratevgood[i]<-vgoodTotal/cellTotal
  }
}

# Plot the vgood cells
plot(som_model,type="property",property=ratevgood,main="VGOOD",palette.name=coolBlueHotRed)
```

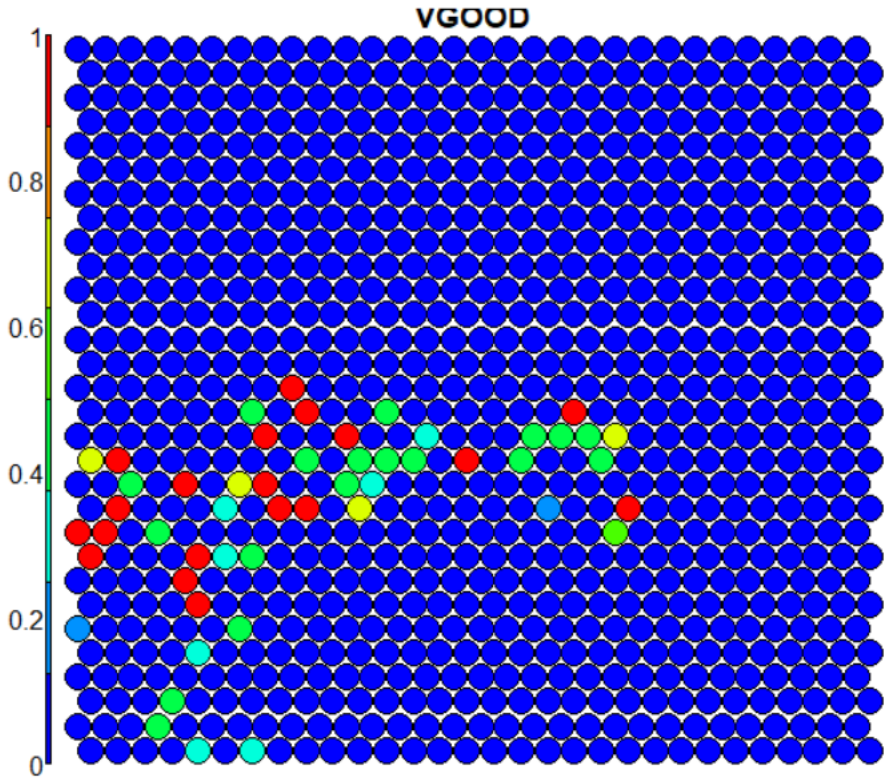

Figure 27. Mapping Plot

So as our main purpose of finding out the very good ‘Ranking’ as it is the most desirable outcome we want, using the given code, we were able to identify the node that gives the very good outcome. In fact, if you compare the Figure 27 with Figure 26, we may be able to identify certain clusters that shares similar traits that is able to produce the very good ‘Ranking’ outcome.

Bayesian Belief Network Analysis

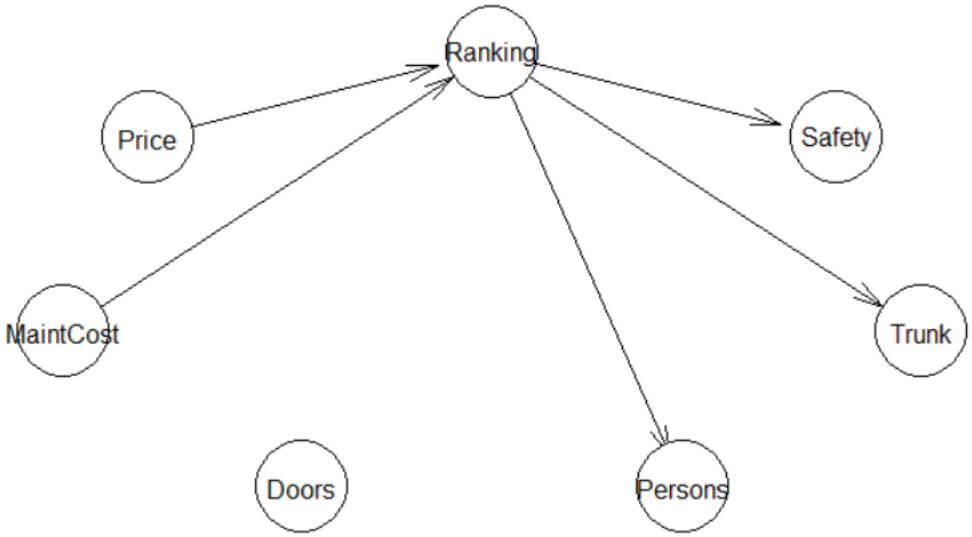

Figure 28. Network Analysis Diagram

The given code for BBN analysis firstly provided us the plot of causality between the variables in our carData set. From the plot, we can observe the BBN concluded that the Ranking variable depends on the variable Trunk and Person in our data set. The Price and Maintenance Cost was observed to be dependent on the Ranking variable instead. The variable Door was deemed not to be of any context causality to the other variables which if we choose to take this analysis would mean we can completely ignore the data of the variable Doors. As we also realize that our target is mainly is the outcome on the variable Ranking, we will find the expected probability for each condition set we are interested in.

If I am a producer of a car and I am going to use this data set to predict on whether I can achieve an optimal product to be sold. As a producer, I will be mainly interested in the price of my car, trunk size, number of person and safety; assuming of course that I don't have to worry about the number of doors as the BBN provides no association of Ranking with this variable. Using these as my set condition, I would be interested mainly in a low-priced car, small trunk size, small number of people and lowest safety features as to reduce my cost of production.

```
> cpquery(bbn,event=(Ranking=="vgood"),evidence=(Safety=='low' & Persons=='2' & Trunk=='small' & Price=="low"))
[1] 0
> cpquery(bbn,event=(Ranking=="good"),evidence=(Safety=='low' & Persons=='2' & Trunk=='small' & Price=="low"))
[1] 0
> cpquery(bbn,event=(Ranking=="acc"),evidence=(Safety=='low' & Persons=='2' & Trunk=='small' & Price=="low"))
[1] 0
> cpquery(bbn,event=(Ranking=="unacc"),evidence=(Safety=='low' & Persons=='2' & Trunk=='small' & Price=="low"))
[1] 1
```

Now from the above queries we did, we can observe that for the cheapest solution to production, I will not be able to produce a car with an acceptable ranking even. To be a more popular amongst user, I decide to let allow for a better safety and a bigger trunk size.

```
> cpquery(bbn,event=(Ranking=="acc"),evidence=(Safety=='med' & Persons=='2' & Trunk=='med' & Price=="low"))
[1] 0
> cpquery(bbn,event=(Ranking=="acc"),evidence=(Safety=='high' & Persons=='2' & Trunk=='high' & Price=="low"))
[1] 0
```

I still did not get an acceptable result. Hence, we kept trying different variables to see if there is a possibility of producing a car at our expected variables.

```
 > cpquery(bbn,event=(Ranking=="acc"),evidence=(Safety=='med' & Trunk=='med'))
[1] 0.3645833
```

For my produced car to even get an acceptable ranking, I must design it as such where the safety feature and trunk must be of medium level.

Now if I were a consumer instead, I would want a car with cheap maintenance, cheap price, big t runk, more number of person, high safety, high ranking.

```
> cpquery(bbn,event=(Ranking=="vgood"),evidence=(Safety=='high' & Persons=='4' & MaintC
ost=='high' & Trunk=='big' & Price=="low"))
[1] 0.5581395
> cpquery(bbn,event=(Ranking=="vgood"),evidence=(Safety=='high' & Persons=='5more' & Ma
intCost=='high' & Trunk=='big' & Price=="low"))
[1] 0
```

From the above queries, I can assume it is possible to find a suitable car for myself if I keep in m ind that it is only possible if I limit the number of persons to 4.