MATH 6350 Fall 2020 MSDS
Homework 1

Sara Nafaryeh
Tony Nguyen
Thomas Su

All authors played equal part

Data Setup

```
> Auto <- read_csv("cleanDataAuto.csv")                #uploading the data into R
> View(Auto)
> Auto = Auto[,-(7:9)]                #rename in order to keep the original safe
                   #---but remove variable columns: “year”, “origin”, and “name”
> names(Auto)
[1] "mpg"  "cylinders"  "displacement" "horsepower"   "weight"    "acceleration"

# For the following report the following variables will be used iner-changably
F1 = cylinders
F2 = displacement
F3 = horsepower
F4 = weight
F5 = acceleration

Response variable = MPG, mpg
```

Part 1

Mean (mF)

```
# Mean for data set Auto for feature variables F1 to F5 and the response variable
MPG
> Auto.mean = colMeans(Auto[,1:6])
> Auto.mean
         mpg    cylinders displacement   horsepower       weight acceleration
   23.445918     5.471939   194.411990   104.469388  2977.584184    15.541327
```

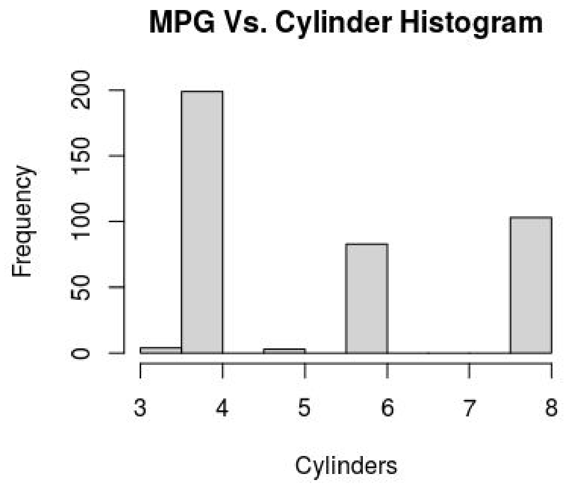
Standard Deviation (stdF)

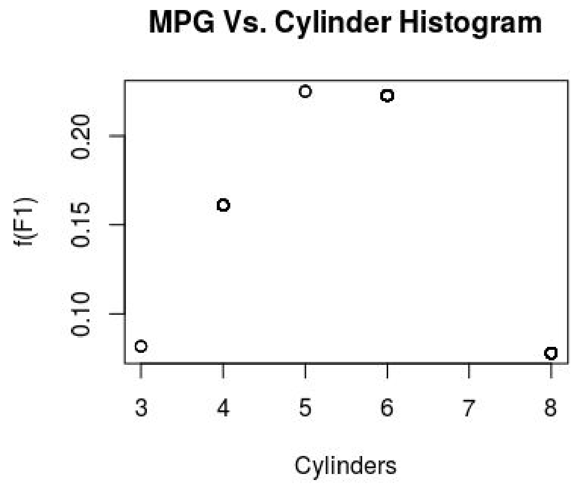
```
# Standard deviation for data set Auto for feature variables F1 to F5 and the
response variable MPG
> for (i in 1: n) {
+  cat(paste0("standard deviation for ", names(Auto[i])), "is", sd(X[,i]),"\n\n")
+ } # standard deviation for X_1 to X_5

standard deviation for mpg is 7.805007                            #response variable
standard deviation for cylinders is 1.705783                      #F1
Standard deviation for displacement is 104.644                    #F2
standard deviation for horsepower is 38.49116                     #F3
standard deviation for weight is 849.4026                         #F4
standard deviation for acceleration is 2.758864                   #F5
```

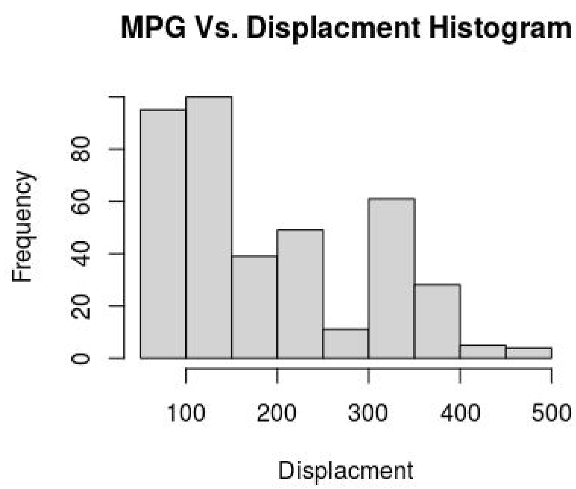
Part 2

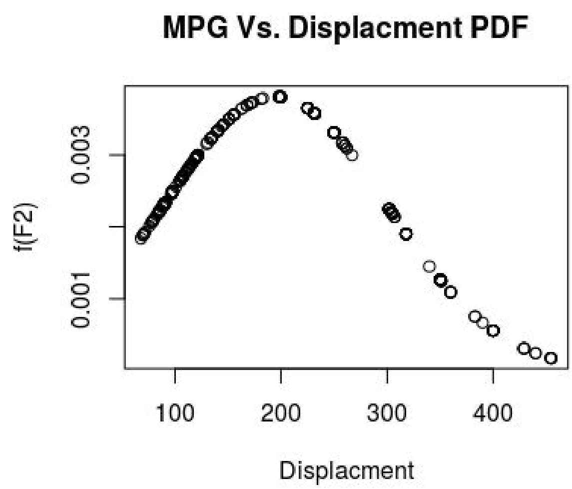
Histogram (histF)

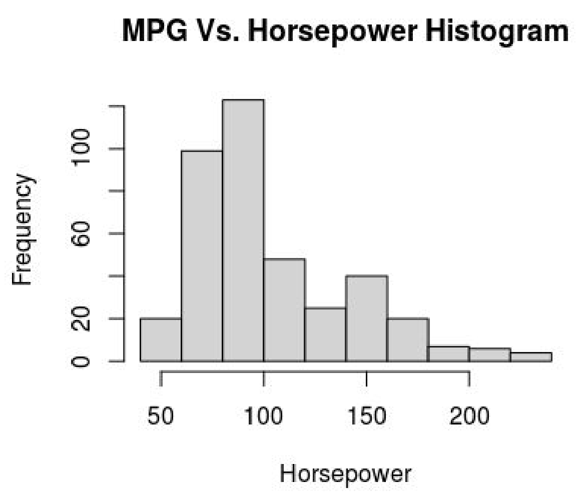
```
##Histogram (with breaks of 10 bins) and Probability Density Function of F1 to F5
par(mfrow = c(1,2))
F1 = Auto$cylinders
hist(F1, xlab = "Cylinders", main = "MPG Vs. Cylinder Histogram", breaks = 10)
plot(F1,dnorm(F1, mean(F1),sd(F1)), xlab = "Cylinders",ylab = "f(F1)", main =
"MPG Vs. Cylinder Histogram")

F2 = Auto$displacement
hist(F2, xlab = "Displacement", main = "MPG Vs. Displacement Histogram", breaks =
10)
plot(F2,dnorm(F2, mean(F2),sd(F2)),xlab = "Displacement", ylab = "f(F2)", main =
"MPG Vs. Displacement PDF")

F3 = Auto$horsepower
hist(F3, xlab = "Horsepower", main = "MPG Vs. Horsepower Histogram", breaks = 10)
plot(F3,dnorm(F3, mean(F3),sd(F3)),xlab = "Horsepower", ylab = "f(F3)", main =
"MPG Vs. Horsepower PDF")

F4 = Auto$weight
hist(F4, xlab = "weight", main = "MPG Vs. weight Histogram", breaks = 10)
plot(F4,dnorm(F4, mean(F4),sd(F4)),xlab = "weight", ylab = "f(F4)", main = "MPG
Vs. weight PDF")

F5 = Auto$acceleration
hist(F5, xlab = "Acceleration", main = "MPG Vs. Acceleration Histogram", breaks =
10)
plot(F5,dnorm(F5, mean(F5),sd(F5)),xlab = "Acceleration", ylab = "f(F5)", main =
"MPG Vs. Acceleration PDF")
par(mfrow = c(1,1))
```

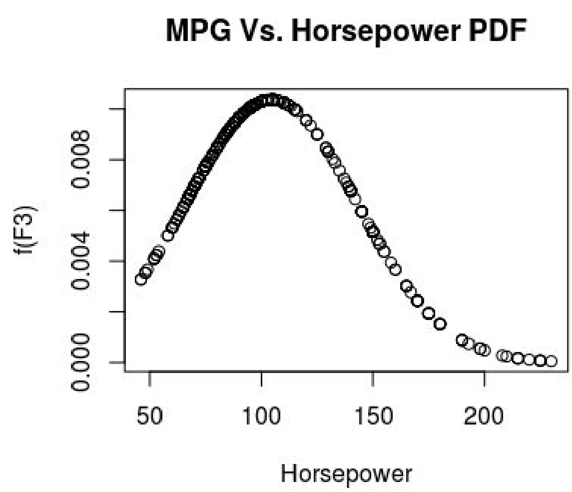

MPG Vs. Cylinder Histogram
Frequency
0
50
100
150
200
3
4
5
6
7
8
Cylinders

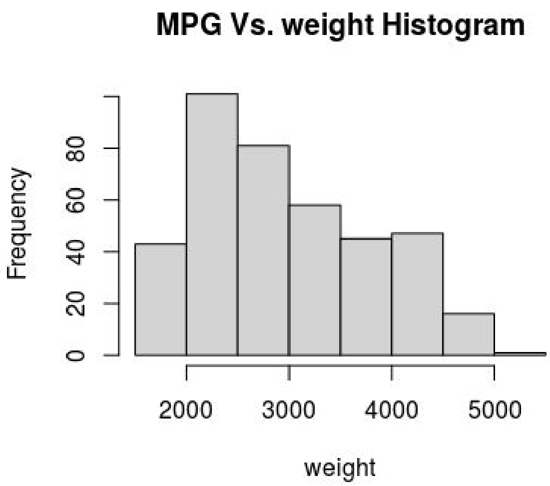

MPG Vs. Cylinder Histogram
f(F1)
0.10
0.15
0.20
3
4
5
6
7
8
Cylinders

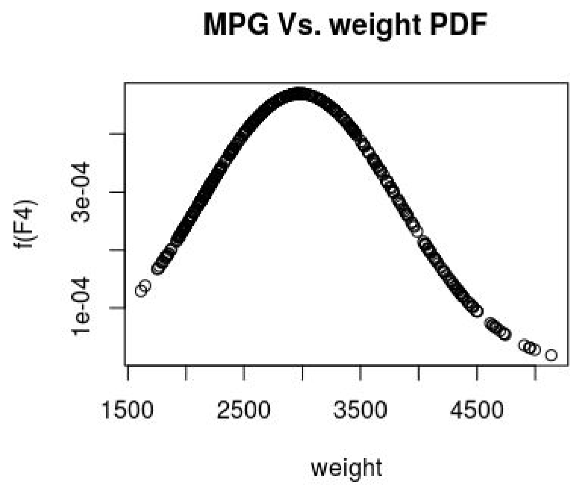

MPG Vs. Displacment Histogram
Frequency
0
20
40
60
80
100
200
300
400
500
Displacment

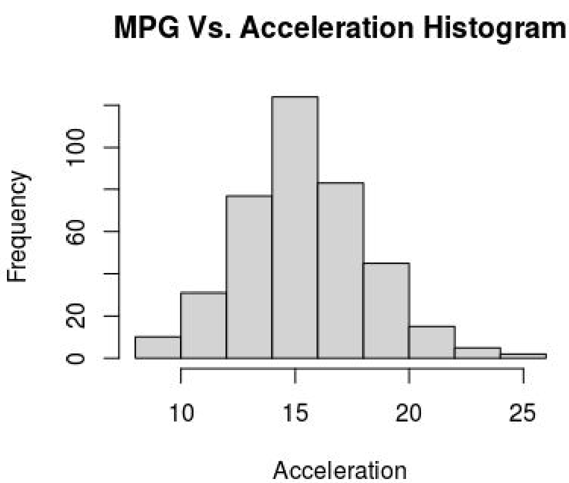

MPG Vs. Displacment PDF
f(F2)
0.001
0.003
100
200
300
400
Displacment

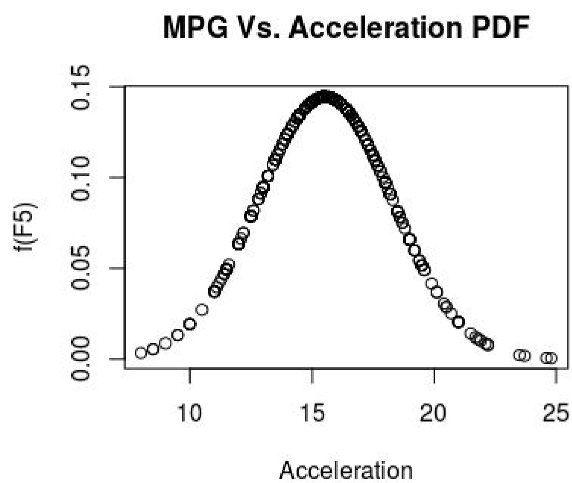

MPG Vs. Horsepower Histogram
Frequency
0
20
60
100
50
100
150
200
Horsepower


MPG Vs. Horsepower PDF
f(F3)
0.000
0.004
0.008
50
100
150
200
Horsepower







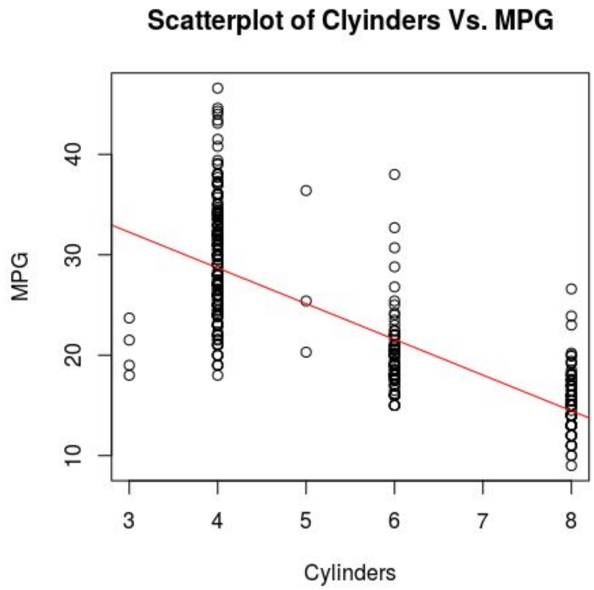


As a result, we observe that Cylinders are mainly distributed in three values of 4,6 and 8. For Displacement, Horsepower, Weight, the distributions are right-skewed which indicates that the mean is larger than the mode of the data for these features. Meanwhile, the distribution of Acceleration looks the most like the probability density function (pdf) of a normal density function with the same mean and standard deviation as F. Thus, it shows that the mean, median, and mode for Acceleration are likely similar.

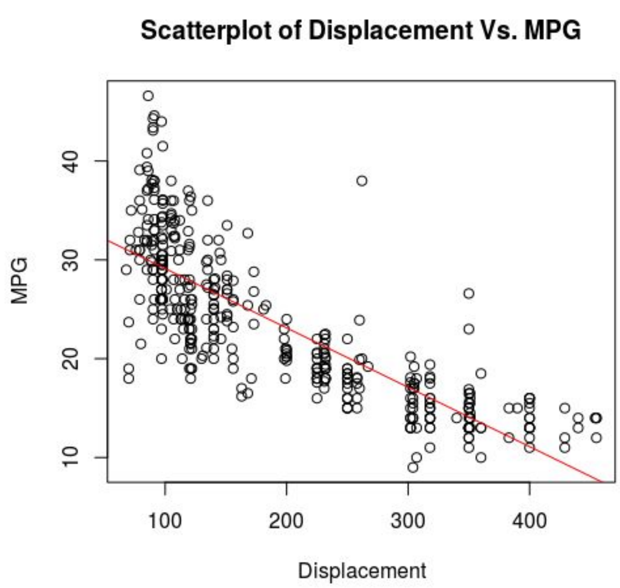
Part 3 + 4

Scatterplots

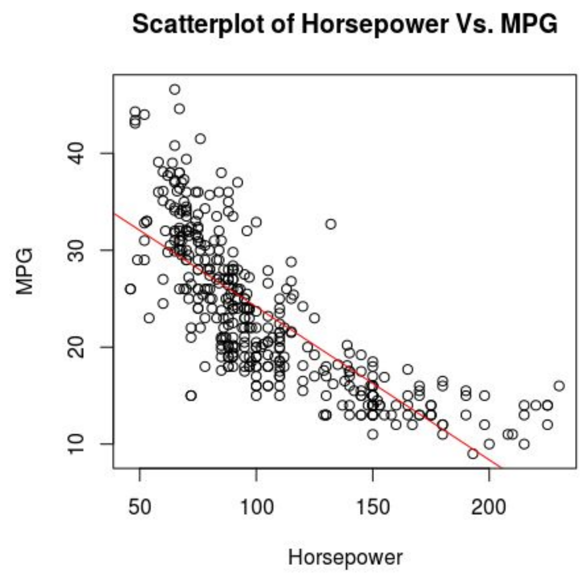
```
# scatterplots of features F1 to F5 respectfully plotted against the response variable mpg and abline to represent the line of the graph

>plot(Auto$cylinders,Auto$mpg, xlab = "Cylinders",
     ylab = "MPG", main = "Scatterplot of Clyinders Vs. MPG")
>abline(lm(mpg~cylinders,data = Auto), col="red")

>plot(Auto$displacement,Auto$mpg, xlab = "Displacement",
     ylab = "MPG", main = "Scatterplot of Displacement Vs. MPG")
>abline(lm(mpg~displacment,data = Auto), col="red")

>plot(Auto$horsepower,Auto$mpg, xlab = "Horsepower",
     ylab = "MPG", main = "Scatterplot of Horsepower Vs. MPG")
>abline(lm(mpg~horsepower,data = Auto), col="red")

>plot(Auto$weight,Auto$mpg, xlab = "Weight",
     ylab = "MPG", main = "Scatterplot of Weight Vs. MPG")
>abline(lm(mpg~weight,data = Auto), col="red")

>plot(Auto$acceleration,Auto$mpg, xlab = "Acceleration",
     ylab = "MPG", main = "Scatterplot of Acceleration Vs. MPG")
>abline(lm(mpg~acceleration,data = Auto), col="red")
```



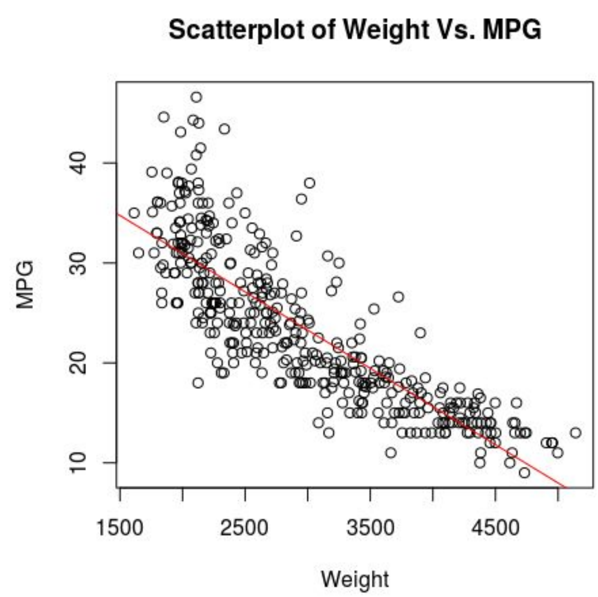
This graph shows Negative Weak linear relationship With MPG along with many outliers. F1 (Cylinders) can also be helpful in predicting mpg, but does not have a strongest capacity to predict the response variable MPG



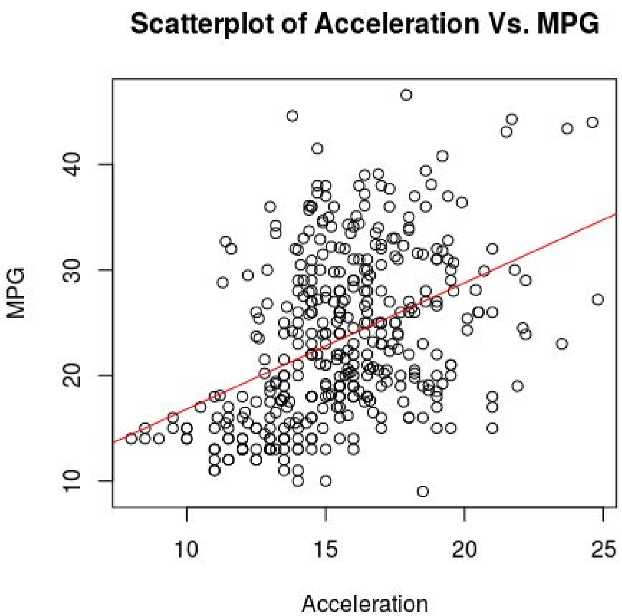
This graph shows Negative Weak linear relationship With MPG along with few outliers. As these features increase, mpg decreases. The form of the relationship also seems linear and the strength of the relationship with mpg seems strong. Therefore, F2 (Displacement) has a strong capacity to predict the response variable MPG. Same reasons are applied for F3 (horsepower) and F4 (Weight).



This graph shows Negative linear relationship With MPG along with few outliers. F3 (Horsepower) demonstrates a higher chance to predict the response variable MPG.



This graph shows Negative linear relationship With MPG along with few outliers. F4 (weight) demonstrates a higher chance to predict the response variable MPG.



This graph shows Positive Weak linear relationship With MPG along with many outliers. Because of the points are more scattered, F5 (Acceleration) does NOT demonstrate a strong capacity to predict the response variable MPG

Part 5

Correlations

```
# Correlation of Data Set Auto from F1 to F5 feature variables

> for (i in 1: p) {
+  cat("Correlation of", names(Auto[i+1]), "Vs. mpg:\n",
+      cor(Auto[, i + 1], Auto[, 1]), "\n\n")
+ }
Correlation of cylinders Vs. mpg:
 -0.7776175

Correlation of displacement Vs. mpg:
 -0.8051269

Correlation of horsepower Vs. mpg:
 -0.7784268

Correlation of weight Vs. mpg:
 -0.8322442

Correlation of acceleration Vs. mpg:
 0.4233285
```

The following data shows the correlations of features F1, F2, F3, F4, F5 vs. the response variable MPG. For F5 (acceleration) we observe a weak positive relationship (correlation) since the correlation is not as high (0.42), thus a weak capacity to predict MPG.

For F1 (cylinders) and F3 (horsepower) we observe a moderate negative relationship (correlation) with MPG thus can predict MPG.

For F2 (displacement) and F4 (weight) we observe a fairly strong negative relationship (correlation) with MPG thus between the two F4 (weight) appears to show a high capacity of predicting MPG.

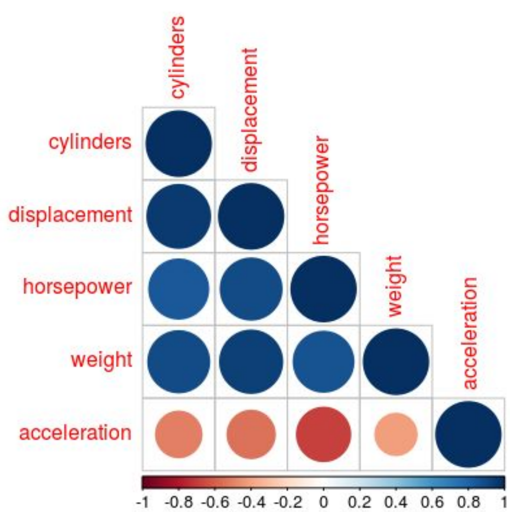
Note: all of these negative relationships (correlations) indicate that when the values in these features increase, mpg decreases and vise-versa, and the correlation values are close to -1 which indicates that they have strong capacity to predict mpg.

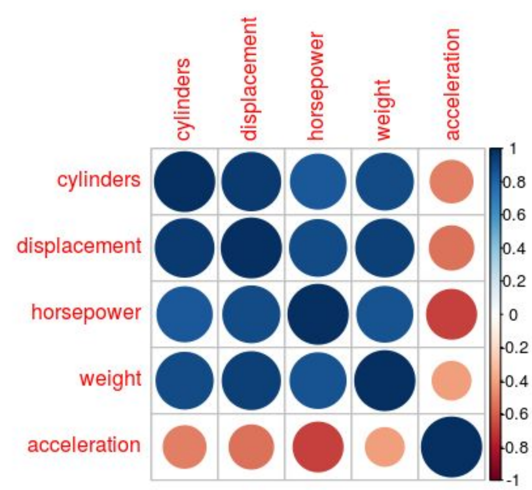
Part 6

Correlation matrix CORR

```
> Car <- Auto                    #renaming from Auto to Car
> cor(Car)
                    mpg  cylinders displacement horsepower     weight acceleration
mpg           1.0000000 -0.7776175   -0.8051269 -0.7784268 -0.8322442    0.4233285
cylinders    -0.7776175  1.0000000    0.9508233  0.8429834  0.8975273   -0.5046834
displacement -0.8051269  0.9508233    1.0000000  0.8972570  0.9329944   -0.5438005
horsepower   -0.7784268  0.8429834    0.8972570  1.0000000  0.8645377   -0.6891955
weight       -0.8322442  0.8975273    0.9329944  0.8645377  1.0000000   -0.4168392
acceleration  0.4233285 -0.5046834   -0.5438005 -0.6891955 -0.4168392    1.0000000
```

```
> library(corrplot)                    # Installing the corrplot package into R
> Auto.Matrix <- data.matrix(Auto[,2:6]) #create a 5x5 matrix of the 5 feature
> Auto.corr <- cor(Auto.Matrix)          # correlation of the 5x5 matrix
> corrplot(Auto.corr,type="lower")       # plotting it
> corrplot(Auto.corr)
```





The following graphs show the 5x5 Correlation Matrix for the 5 features F1,F2,F3,F4,F5. The legend on the bottom and left explain the correlation coefficient based on color: positive correlation displayed in blue and negative correlations displayed in red. Meanwhile the size of the circles demonstrates the strength of the correlation coefficient: larger the circle demonstrates a stronger correlation and smaller the circle demonstrates weaker correlation. Note: some of these features are strongly correlated with each other such as cylinders and displacement, and displacement and weight. They have correlation values in the 0.9 range. As a result, based on the correlation matrix mpg has a strong negative correlation with cylinders, displacement, horsepower, and weight. MPG has a positive correlation with acceleration, but the strength of the relationship is weak.

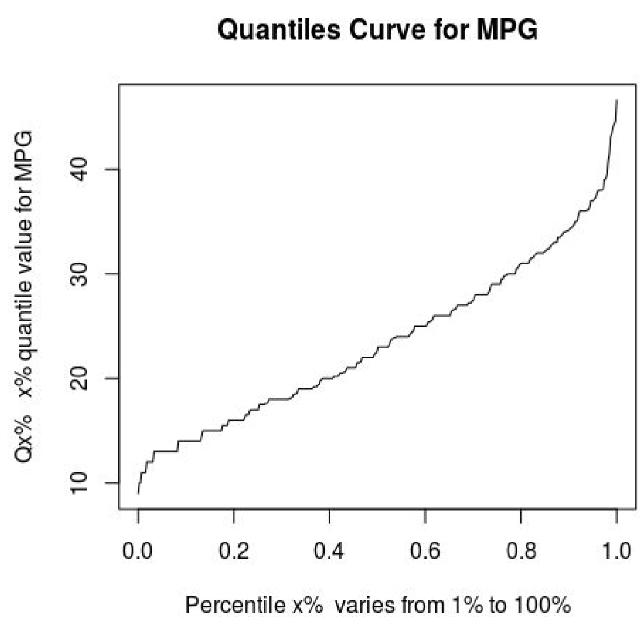
Part 7

Quantile Curve

```
> mpg = data.matrix(Auto[,1])
> quantile = quantile(mpg, probs = seq(0, 1, by = 0.01))
> print(quantile)
    0%     1%     2%     3%     4%     5%     6%     7%     8%     9%    10%    11%    12%    13%    14% 
 9.000 11.000 12.000 12.000 13.000 13.000 13.000 13.000 13.000 14.000 14.000 14.000 14.000 14.000 15.000 
   15%    16%    17%    18%    19%    20%    21%    22%    23%    24%    25%    26%    27%    28%    29% 
15.000 15.000 15.000 15.500 16.000 16.000 16.000 16.004 16.500 17.000 17.000 17.500 17.657 18.000 18.000 
   30%    31%    32%    33%    34%    35%    36%    37%    38%    39%    40%    41%    42%    43%    44% 
18.000 18.000 18.112 18.503 19.000 19.000 19.000 19.200 19.632 20.000 20.000 20.200 20.344 20.600 21.000 
   45%    46%    47%    48%    49%    50%    51%    52%    53%    54%    55%    56%    57%    58%    59% 
21.000 21.500 22.000 22.000 22.000 22.750 23.000 23.000 23.723 24.000 24.000 24.000 24.287 25.000 25.000 
   60%    61%    62%    63%    64%    65%    66%    67%    68%    69%    70%    71%    72%    73%    74% 
25.000 25.451 26.000 26.000 26.000 26.000 26.600 27.000 27.000 27.158 27.470 28.000 28.000 28.043 29.000 
   75%    76%    77%    78%    79%    80%    81%    82%    83%    84%    85%    86%    87%    88%    89% 
29.000 29.500 29.907 30.000 30.445 30.980 31.000 31.424 31.853 32.000 32.135 32.478 33.000 33.500 33.998 
   90%    91%    92%    93%    94%    95%    96%    97%    98%    99%   100% 
34.190 34.662 35.532 36.000 36.100 37.000 37.808 38.027 39.652 43.454 46.600 
```

```
> x =Auto$mpg
> n = length(x)
> plot((1:n - 1)/(n - 1),
sort(x), type="l",  main =
"Quantiles Curve for MPG", xlab =
"Percentile x%  varies from 1% to
100%", ylab = "Qx%   x% quantile
value for MPG")
```

The following code and graph demonstrates the quantile curve for the response
variable MPG 0% to 100%



Part 8

Low vs. High MPG Quantile

```
> Q33 <- quantile(Auto$mpg, probs = .33) # find the mpg quantile of 33%
> Q66 <- quantile(Auto$mpg, probs = .66) # find the mpg quantile of 66%

> LOWmpg <- Auto$mpg(Auto$mpg <= Q33)    #gather mpg below and equal 33% quantile
> HIGHmpg<- Auto$mpg(Auto$mpg > Q66)     #gather mpg above 66% quantile

> LOWmpg.table <- Auto[mpg <= Q33,]      # create a table of LOW.mpg quantile
> HIGHmpg.table <- Auto[mpg > Q66,]      #create a table of the HIGH.mpg quantile
```

In the following R code, two disjoint tables of cases are extracted.
LOWmpg, we identify the mpg whose quantile probability is from 33% (Q33) and below of case features F1:F5, while HIGHmpg identifies the mpg quantile probability of 66% (Q66) and higher of case features F1:F5. They are then gathered into tables of LOW and HIGH MPG respectfully.

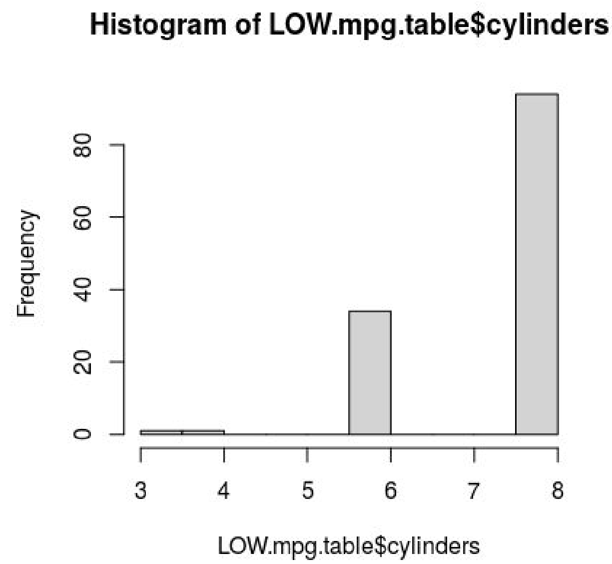
```
> head(HIGHmpg.table)
# A tibble: 6 x 6
    mpg cylinders displacement horsepower weight acceleration
  <dbl>     <dbl>        <dbl>      <dbl>  <dbl>        <dbl>
    27         4           97         88   2130         14.5
    27         4           97         88   2130         14.5
    28         4          140         90   2264         15.5
    28         4          116         90   2123         14
    30         4           79         70   2074         19.5
    30         4           88         76   2065         14.5
> head(LOWmpg.table)
# A tibble: 6 x 6
    mpg cylinders displacement horsepower weight acceleration
  <dbl>     <dbl>        <dbl>      <dbl>  <dbl>        <dbl>
    18         8          307        130   3504         12
    15         8          350        165   3693         11.5
    18         8          318        150   3436         11
    16         8          304        150   3433         12
    17         8          302        140   3449         10.5
    15         8          429        198   4341         10
```

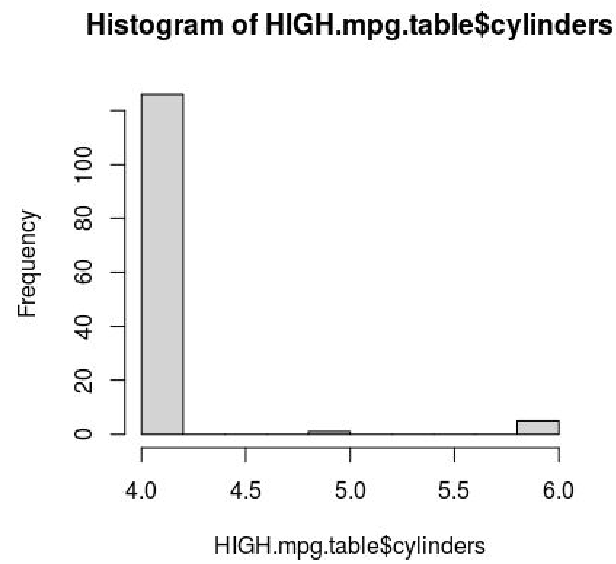
Part 9 + 10

Histogram of LOWmpg and HIGHmpg

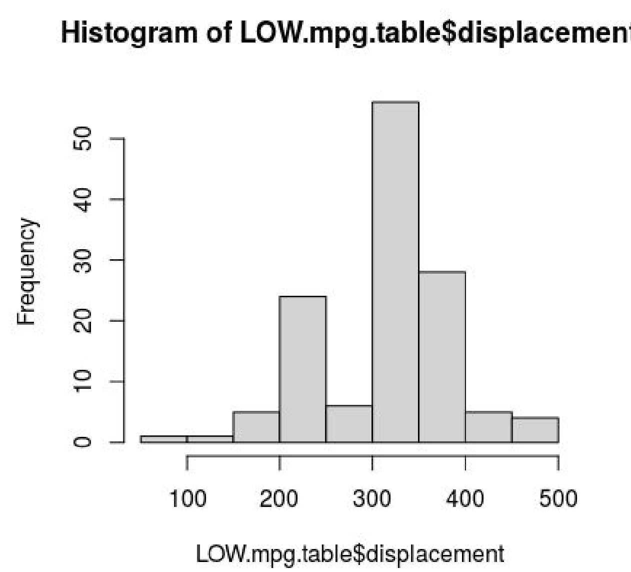
```
# With Bins of 10

par(mfrow=c(1,2))
##histogram for F1 LOW / HIGH mpg
hist(LOWmpg.table$cylinders, breaks = 10)
hist(HIGHmpg.table$cylinders, breaks = 10)

##histogram for F2 LOW / HIGH mpg
hist(LOWmpg.table$displacement, breaks = 10)
hist(HIGHmpg.table$displacement, breaks = 10)

##histogram for F3 LOW / HIGH mpg
hist(LOWmpg.table$horsepower, breaks = 10)
hist(HIGHmpg.table$horsepower, breaks = 10)

##histogram for F4 LOW / HIGH mpg
hist(LOWmpg.table$weight, breaks = 10)
hist(HIGHmpg.table$weight, breaks = 10)

##histogram for F5 LOW / HIGH mpg
hist(LOWmpg.table$acceleration, breaks = 10)
hist(HIGHmpg.table$acceleration, breaks = 10)
```

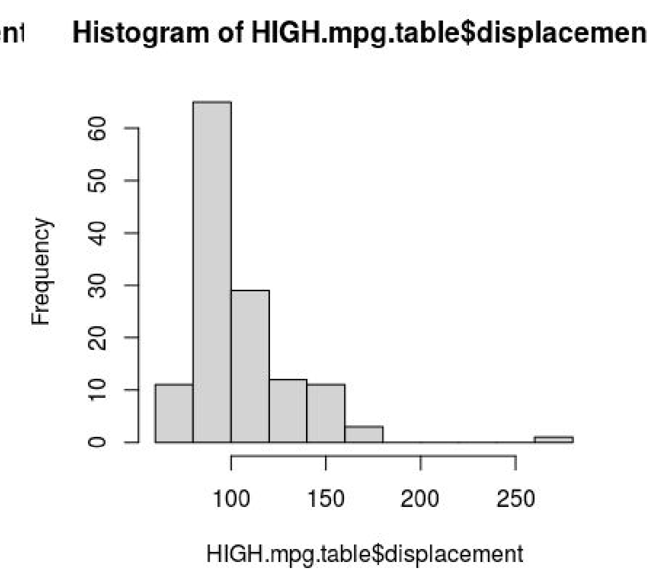
```
> dim(LOWmpg.table)
[1] 130   6
> dim(HIGHmpg.table)
[1] 132   6
```



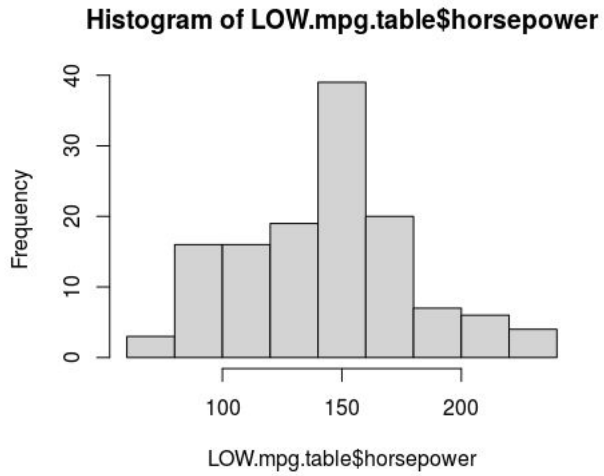


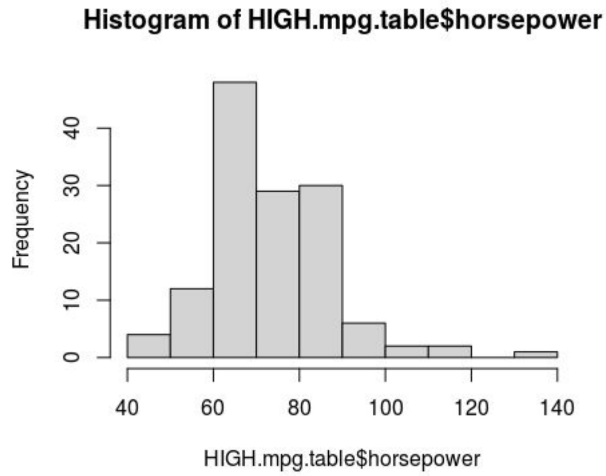
Histogram of LOW mpg for cylinders shows range from 3 to 8 of 10 bins, peak at 8 cylinders, and skewed to the left. Also appear to have outliers between 3 and 4.
Histogram of HIGH mgp also shows range from 3 to 6 of 10 bins, but a peak at 4 cylinders, skewed to the right.



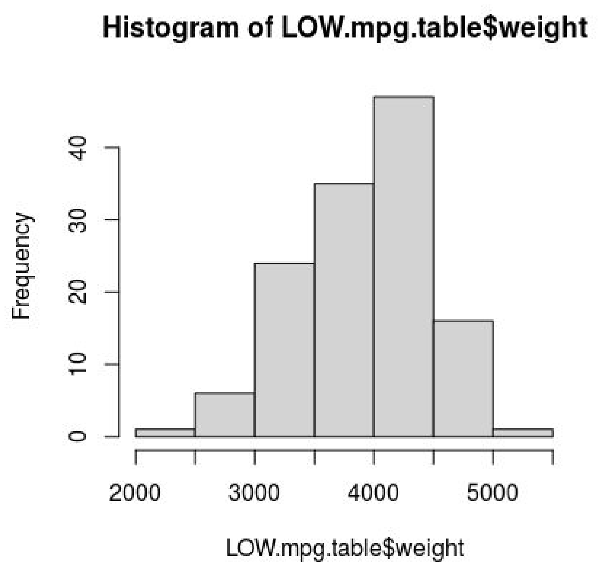


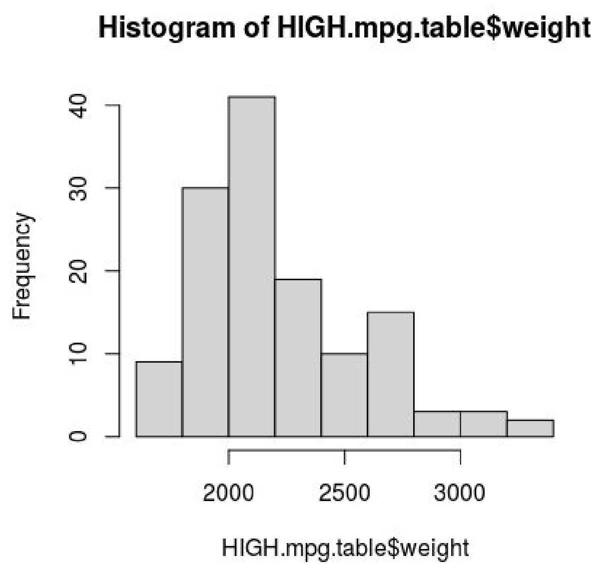
Histogram of LOW mpg for displacement shows range from 50 to 500, peak between 300-350, with undefined shape. HIGH mpg of displacement shows range from 0 to 300, peak between 50-100 with shape right skewed.



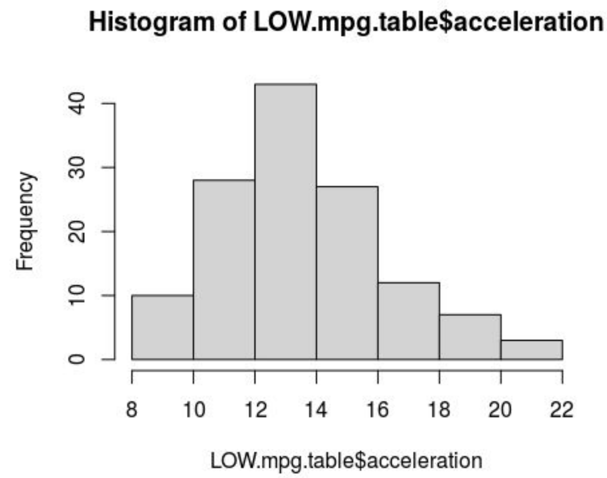


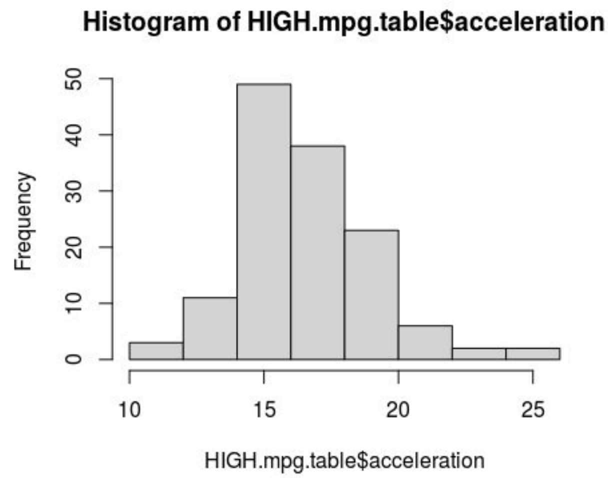
Histogram of LOW mpg for horsepower shows range from 0 to 300, peak at 150, with undefined shape (but close to bell shape). HIGH mpg of horsepower shows range from 40 to 140, peak between 60-70 with undefined shape (right skewed).





Histogram of LOW mpg for weight shows range from 2000 to 5500, peak between 4000 to 4500, with bell shape/ left skewed. HIGH mpg of weight shows range from 1000 to 4000, peak between 2000-2250 with shape right skewed.





Histogram of LOW mpg for acceleration shows range from 8 to 22, peak between 12-14, with bell shaped. HIGH mpg of acceleration shows range from 10-25, peak at 15 with bell shape/ skewed right. Thus, some overlap for the distribution of acceleration between high and low mpg cars, between 10 and 22 which could cause errors later if using acceleration to classify between low and high mpg cars.

As a conclusion, we can observe that Cylinders, Displacement, Horsepower, and Weight vary greatly between low mpg cars and high mpg cars. Therefore, Cylinders, Displacement, Horsepower, and Weight (not including Acceleration) may have a good capacity to discriminate between high mpg and low mpg.

Part 11

Mean and SD of LOWmpg and HIGHmpg

```
> # 11
> # reminder that X = data.matrix(Auto[, 2: ncol(Auto)])
> #                 mpg = data.matrix(Auto[,1])
> #                 LOWmpg.table = X[mpg <= quantile(mpg, probs = c(0.33)),]
> #                 HIGHmpg.table = X[mpg > quantile(mpg, probs = c(0.66)),]
>
> mL = function(i) {mean(LOWmpg.table[, i])}
> stdL = function(i) {sd(LOWmpg.table[, i])}
> mH = function(i) {mean(HIGHmpg.table[, i])}
> stdH = function(i) {sd(HIGHmpg.table[, i])}
>
> for (i in 1: p) {
+     cat(paste0("mean for ", names(Auto[i + 1]),
+                " associated to all cases with LOWmpg is\n",
+                mL(i),"\n\n"))
+ }
mean for cylinders associated to all cases with LOWmpg is
7.40769230769231

mean for displacement associated to all cases with LOWmpg is
315.307692307692

mean for horsepower associated to all cases with LOWmpg is
145.623076923077

mean for weight associated to all cases with LOWmpg is
3937.33076923077

mean for acceleration associated to all cases with LOWmpg is
13.7784615384615
>
> for (i in 1: p) {
+     cat(paste0("standard deviation for ", names(Auto[i + 1]),
+                " associated to all cases with LOWmpg is\n",
+                stdL(i),"\n\n"))
+ }
standard deviation for cylinders associated to all cases with LOWmpg is
1.00922991477195

standard deviation for displacement associated to all cases with LOWmpg is
71.1140416311286

standard deviation for horsepower associated to all cases with LOWmpg is
```

```
35.8410066530087

standard deviation for weight associated to all cases with LOWmpg is
557.18569390781

standard deviation for acceleration associated to all cases with LOWmpg is
2.64929373696289
>
> for (i in 1: p) {
+     cat(paste0("mean for ", names(Auto[i + 1]),
+                " associated to all cases with HIGHmpg is\n",
+                mH(i),"\n\n"))
+ }
mean for cylinders associated to all cases with HIGHmpg is
4.08333333333333

mean for displacement associated to all cases with HIGHmpg is
106.401515151515

mean for horsepower associated to all cases with HIGHmpg is
74.3939393939394

mean for weight associated to all cases with HIGHmpg is
2226.09090909091

mean for acceleration associated to all cases with HIGHmpg is
16.5606060606061
>
> for (i in 1: p) {
+     cat(paste0("standard deviation for ", names(Auto[i + 1]),
+                " associated to all cases with HIGHmpg is\n",
+                stdH(i),"\n\n"))
+ }
standard deviation for cylinders associated to all cases with HIGHmpg is
0.391545512093693

standard deviation for displacement associated to all cases with HIGHmpg is
26.4222494285094

standard deviation for horsepower associated to all cases with HIGHmpg is
13.8843409933899

standard deviation for weight associated to all cases with HIGHmpg is
345.877851062562

standard deviation for acceleration associated to all cases with HIGHmpg is
2.51998902141549
```

Part 12

Discrimination Power of each feature F1 to F5

```
#  N = nrow(LOWmpg.table) = 130

> mL = matrix(colMeans(LOWmpg.table))
> mH = matrix(colMeans(HIGHmpg.table))
> stdL = matrix(apply(LOWmpg.table, MARGIN = 2, sd))
> stdH = matrix(apply(HIGHmpg.table, MARGIN = 2, sd))
> s = function(F) sqrt((stdL^2 + stdH^2) / nrow(LOWmpg.table))[F]
> discr = function(F) abs(mH[F,1] - mL[F,1]) / s(F)
>
> for (i in 1 : 5) {
+     print(s(i))
+ }
[1] 0.09494342
[1] 6.65371
[1] 3.371091
[1] 57.51838
[1] 0.3206856
>
> for (i in 1 : 5) {
+     print(discr(i))
+ }
[1] 35.01411
[1] 31.39694
[1] 21.12941
[1] 29.75118
[1] 8.675614
```

Discriminating power of feature F1 to F5 between LOW and HIGH mpg
F1 (cylinders) appears to have the largest discriminating power of 35.0141, but displacement and weight are following very closely. Horsepower is the next lower while F5(acceleration) has the smallest discrimination power of 8.6756. As a result, to distinguish between low and high mpg cars, we would better choose cylinders, displacement, and weight based on the discriminating powers.

Part 13

Threshold of each feature F1 to F5

```
> thr = (mL * stdH + mH * stdL) / (stdH + stdL)
> print(thr)
           [,1]
[1,]    5.01256
[2,]  162.99349
[3,]   94.28258
[4,] 2881.50433
[5,]   15.20433
```

The following code used to compute for each case #n, a scoreF(n) based on the value F(n), as follows:
when the feature F verifies mH > mL, then
scoreF(n) = 1 , if F(n) > thrF
scoreF(n) = -1 , if F(n) <= thrF
when the feature F verifies mH < mL, then
scoreF(n) = 1 , if F(n) < thrF
scoreF(n) = -1 , if F(n) >= thrF

By observation it is clear that:
mH > mL is feature F5 and mH < mL are features F1:F4

```
> # reminder that X = data.matrix(Auto[, 2: ncol(Auto)])
> scoreF = function(j) {
+     score = c()
+     if (mH[j,] > mL[j,]) {
+         for (i in 1: nrow(X)) {
+             if (X[i, j] > thr[j, ]) {
+                 score[i] = 1} else {
+                     score[i] = -1
+                 }
+         }
+     } else {
+         for (i in 1: nrow(X)) {
+             if (X[i, j] < thr[j, ]) {
+                 score[i] =  1} else {
+                     score[i] = - 1
+                 }
+         }
+     }
+     print(score)
+ }
```

```
> scoreF(1)
  [1] -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1  1 -1 -1 -1  1  1  1  1  1  1 -1
-1 -1 -1 -1  1  1  1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1  1 -1 -1  1  1
 [51]  1  1  1  1  1  1  1  1  1  1  1 -1 -1 -1 -1 -1 -1 -1 -1 -1  1 -1 -1 -1 -1
1  1  1  1  1  1  1  1  1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
[101] -1  1 -1 -1 -1 -1 -1  1  1  1  1  1 -1  1 -1 -1  1  1  1  1 -1  1 -1 -1 -1
-1 -1  1  1  1  1 -1 -1 -1 -1 -1 -1 -1 -1  1  1  1  1  1  1  1  1  1  1  1
[151] -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1  1  1 -1  1  1  1  1 -1  1 -1
1  1  1  1  1  1  1  1  1  1 -1 -1 -1 -1 -1 -1 -1 -1  1  1  1  1 -1 -1 -1
[201] -1  1  1  1  1  1 -1  1 -1 -1 -1 -1 -1 -1  1  1  1  1  1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1  1  1  1  1  1  1  1  1 -1  1  1  1  1  1  1  1 -1 -1 -1
[251] -1 -1 -1  1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1  1  1  1  1  1  1  1  1  1 -1  1
-1  1  1 -1 -1  1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1  1  1  1  1  1 -1  1 -1  1
[301]  1  1  1  1 -1 -1  1  1  1  1  1  1  1  1 -1  1  1  1  1  1  1  1  1  1  1
1  1  1  1  1 -1  1  1  1  1  1  1 -1  1  1  1  1  1  1  1  1  1  1  1  1
[351]  1  1  1  1  1 -1 -1 -1 -1 -1 -1 -1  1  1  1  1  1  1  1  1  1  1  1  1  1
1  1  1  1  1 -1 -1  1 -1  1  1  1  1  1  1  1  1
> scoreF(2)
  [1] -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1  1 -1 -1 -1  1  1  1  1  1  1 -1
-1 -1 -1 -1  1  1  1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1  1 -1 -1  1  1
 [51]  1  1  1  1  1  1  1  1  1  1  1 -1 -1 -1 -1 -1 -1 -1 -1 -1  1 -1 -1 -1 -1
1  1  1  1  1  1  1  1  1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
[101] -1  1 -1 -1 -1 -1 -1  1  1  1  1  1  1  1 -1 -1  1  1  1  1 -1  1  1 -1 -1
-1 -1  1  1  1  1 -1 -1 -1 -1 -1 -1 -1 -1  1  1  1  1  1  1  1  1  1  1  1
[151] -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1  1  1 -1  1  1  1  1 -1  1 -1
1  1  1  1  1  1  1  1  1  1 -1 -1 -1 -1 -1 -1 -1 -1  1  1  1  1 -1 -1 -1
[201] -1  1  1  1  1  1 -1  1  1 -1 -1 -1 -1 -1  1  1  1  1  1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1 -1 -1 -1
[251] -1 -1 -1  1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1  1  1  1  1  1  1  1  1  1 -1  1
-1  1  1 -1 -1  1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1  1  1  1  1 -1 -1  1 -1  1
[301]  1  1  1  1 -1 -1  1  1  1  1  1  1  1  1 -1  1  1  1  1  1  1  1  1  1  1
1  1  1  1  1 -1  1  1  1  1  1  1 -1  1  1  1  1  1  1  1  1  1  1  1  1
[351]  1  1  1  1  1  1 -1  1 -1 -1 -1 -1  1  1  1  1  1  1  1  1  1  1  1  1  1
1  1  1  1  1 -1 -1  1 -1  1  1  1  1  1  1  1  1
> scoreF(3)
  [1] -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1  1  1  1  1  1 -1 -1  1
-1 -1 -1 -1  1  1 -1 -1 -1 -1  1 -1 -1 -1 -1 -1 -1 -1 -1 -1  1 -1  1  1  1
 [51]  1  1  1  1  1  1 -1  1  1  1  1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1  1  1  1  1  1 -1  1  1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1  1
[101] -1  1 -1 -1 -1 -1 -1  1  1  1  1  1 -1  1 -1 -1  1  1  1 -1 -1 -1 -1 -1 -1
-1 -1  1  1  1  1 -1 -1 -1 -1 -1 -1 -1 -1  1  1  1  1  1  1  1  1 -1  1  1
[151] -1 -1  1  1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1  1  1 -1  1 -1  1 -1 -1  1  1
-1  1 -1 -1  1  1  1  1  1  1 -1 -1 -1 -1 -1 -1  1  1  1  1  1  1 -1  1 -1
[201] -1  1  1  1  1 -1 -1  1 -1 -1 -1 -1 -1 -1  1  1  1 -1  1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1  1  1  1  1  1  1  1  1 -1 -1 -1  1  1  1  1  1 -1 -1 -1
[251] -1 -1  1  1 -1  1 -1  1 -1 -1 -1 -1 -1 -1  1 -1 -1  1 -1 -1  1 -1 -1 -1 -1
-1  1  1 -1  1  1  1 -1 -1 -1 -1 -1 -1 -1 -1 -1  1  1  1  1  1 -1  1  1  1
```

```
[301]  1  1  1  1 -1 -1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1 -1  1  1  1
1  1  1  1  1 -1 -1  1  1  1  1  1 -1  1  1  1  1  1  1  1  1  1  1  1  1  1
[351]  1  1 -1  1  1  1 -1 -1 -1 -1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
1  1  1  1  1 -1  1  1 -1 -1  1  1  1  1  1  1  1
> scoreF(4)
  [1] -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1  1  1  1  1  1  1  1  1  1  1  1
-1 -1 -1 -1  1  1  1  1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1  1 -1 -1  1  1
 [51]  1  1  1  1  1  1  1  1  1  1  1 -1 -1 -1 -1 -1 -1 -1 -1 -1  1 -1 -1 -1 -1
-1  1 -1  1  1  1  1  1  1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
[101] -1  1 -1 -1 -1 -1  1  1  1  1  1  1  1  1 -1 -1  1  1  1  1 -1  1  1 -1 -1
-1 -1  1  1  1  1 -1 -1 -1 -1 -1 -1 -1 -1  1  1  1  1  1  1  1  1  1  1  1
[151] -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1  1  1 -1  1  1  1  1 -1  1 -1
1 -1 -1  1  1  1  1  1  1  1 -1 -1 -1 -1 -1 -1 -1 -1  1  1  1  1 -1 -1 -1
[201] -1  1  1  1  1 -1 -1 -1 -1 -1 -1 -1 -1 -1  1  1  1  1  1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1 -1 -1 -1
[251] -1 -1 -1  1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1  1  1  1  1  1  1  1  1  1 -1  1
-1  1  1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1  1  1  1  1 -1 -1 -1 -1  1
[301]  1  1  1  1  1  1  1  1  1  1  1  1  1 -1 -1  1  1  1  1  1  1  1  1  1  1
-1 -1  1  1  1 -1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
[351]  1  1  1  1 -1 -1 -1 -1 -1 -1 -1 -1  1  1  1  1  1  1  1  1  1  1  1  1  1
1  1  1  1  1 -1 -1  1  1  1  1 -1  1  1  1  1  1
> scoreF(5)
  [1] -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1  1  1  1 -1  1  1 -1  1 -1 -1
-1 -1 -1  1 -1  1 -1 -1  1  1  1  1 -1 -1 -1 -1 -1 -1 -1 -1  1 -1 -1 -1 -1
 [51]  1 -1  1  1  1  1  1  1  1  1  1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1  1 -1
-1  1  1  1  1  1 -1 -1  1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1  1  1  1  1
[101]  1  1 -1 -1 -1 -1 -1  1  1  1 -1  1 -1  1 -1 -1  1  1 -1  1 -1 -1 -1 -1  1
1  1  1  1  1  1  1  1  1 -1 -1 -1  1  1  1  1 -1  1  1 -1  1 -1 -1  1  1
[151]  1  1  1  1 -1 -1 -1 -1  1  1  1  1 -1 -1 -1  1  1  1  1 -1  1  1 -1 -1  1
-1  1 -1 -1  1  1  1 -1  1  1 -1 -1 -1 -1  1 -1  1  1  1  1 -1  1  1  1  1
[201]  1 -1  1  1 -1  1 -1  1  1  1 -1 -1 -1 -1  1 -1  1  1  1 -1  1 -1 -1  1  1
1  1 -1 -1 -1 -1 -1  1  1  1  1  1  1 -1 -1 -1 -1  1 -1  1  1  1  1 -1 -1
[251]  1  1  1  1  1  1  1  1  1 -1 -1 -1 -1 -1  1 -1 -1 -1 -1  1  1 -1  1 -1  1
1 -1  1  1  1  1  1  1  1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1  1  1  1  1 -1
[301] -1  1 -1  1 -1 -1 -1 -1  1  1  1  1  1  1  1  1  1  1 -1 -1  1 -1  1  1  1
1  1 -1  1  1 -1 -1 -1  1  1  1 -1 -1 -1  1  1  1  1  1  1  1 -1  1  1 -1
[351] -1  1 -1  1  1  1 -1 -1  1  1  1  1  1  1  1  1  1  1  1  1  1  1 -1  1 -1
-1  1 -1  1  1  1  1 -1 -1 -1 -1  1  1  1 -1  1  1
```

Part 14

1. We compute compute the score.full(n) for each case n by adding score.F1(n) + score.F2(n) + score.F3(n) + score.F4(n) + score.F5(n) and obtain the result as below:

```
> Car <- Auto                    #renaming from Auto to Car for the rest of part 14
> newCar = data.frame(Car, ScoreF(1), ScoreF(2), ScoreF(3), ScoreF(4), ScoreF(5))
>
> score.full = rep(0, length(Car$mpg))
> score.full <- rowSums(newCar[, c(7, 8, 9, 10, 11)])
> score.full
  [1] -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5  1 -1 -1  1  3  5  5  3  3  1 -1 -5
-5 -5 -3  3  5  1 -3 -3 -3 -1 -3 -5
 [39] -5 -5 -5 -5 -5 -5 -5  5 -5 -3  3  3  5  3  5  5  5  5  3  5  5  5  5 -5 -5 -5
-5 -5 -5 -5 -5 -5  1 -5 -5 -3 -5 -1
 [77]  5  3  5  5  5  1  3  5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -3 -3 -3 -1 -3  5
-5 -5 -5 -5 -3  5  5  5  3  5 -1  5
[115] -5 -5  5  5  3  3 -5  1 -1 -5 -3 -3 -3  5  5  5  5 -3 -3 -3 -5 -5 -5 -3 -3  5
5  3  5  5  3  5  3  1  5  5 -3 -3
[153] -1 -1 -5 -5 -5 -5 -3 -3 -3 -3 -5 -5 -5  5  5 -3  5  1  5  3 -5  3 -1  1  3 -1
1  5  5  5  3  5  5 -5 -5 -5 -5 -3
[191] -5 -1 -1  5  5  3  5 -3 -1 -3 -3  3  5  5  3  1 -5  3 -1 -3 -5 -5 -5 -5  5  3
5  3  5 -5 -3 -5 -5 -3 -3 -3 -3 -5
[229] -5 -5 -5  3  5  5  5  5  5  5  3 -1  1  1  5  3  5  5  5 -3 -5 -5 -3 -3 -1  5
-3 -1 -3 -1 -3 -5 -5 -5 -5 -5  5  1
[267]  1  3  1  3  5  1  3 -5  3 -3  3  5 -3 -1  3 -1 -3 -3 -5 -5 -5 -5 -5 -5 -5  3
3  3  3  1 -3  3 -1  3  3  5  3  5
[305] -3 -3  3  3  5  5  5  5  5  3 -1  5  5  5  3  3  5  1  5  5  5  3  3  3  5  5
-5  1  3  5  5  5  3 -3  3  5  5  5
[343]  5  5  5  5  3  5  5  3  3  5  1  5  3  1 -5 -3 -3 -3 -1 -1  5  5  5  5  5  5
5  5  5  5  3  5  3  3  5  3  5  5
[381] -3 -1  3 -3  1  3  3  5  5  3  5  5
```

The score.full(n) contains both negative and positive numeric values which ranges from -5 to 5.

2. We use the fixing A = 1 (A can be 0, 1, 2 in this case), and define the classifier Pred_A1 associated to the threshold A as follows, for any case "n", compute score.full(n):
   If *score.full(n)* < *A*, Pred_A1 decides that case n has low mpg denoted as 0
   If *score.full(n)* >= *A*, Pred_A1 decides that case n has high mpg denoted as 1

```
> Pred_A1 = rep(0, length(Car$mpg))
> Pred_A1[score.full >= 1] = 1
> Pred_A1
  [1] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 1 1 1 1 1 1 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0
0 0 0 0 0 1 0 0 1 1 1 1 1 1 1 1 1
 [58] 1 1 1 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 1 0 0 0 0 0 1 1 1 1 1 0 1
[115] 0 0 1 1 1 1 0 1 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 1 1
[172] 1 0 1 0 1 1 0 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 1 0 1 0 0 0
0 0 0 1 1 1 1 1 0 0 0 0 0 0 0 0 0
[229] 0 0 0 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 1 1 1
1 1 1 1 1 0 1 0 1 1 0 0 1 0 0 0 0
[286] 0 0 0 0 0 0 1 1 1 1 1 0 1 0 1 1 1 1 1 0 0 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 0 1 1 1 1 1 1 0 1 1 1 1
[343] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0
1 0 1 1 1 1 1 1 1 1
```

3. We compute med = median(all mpg values) = 50% quantile of all mpg values on the whole set of cases. Wee define *the* truePRE class for all cases as follows:
   If *mpg(n)* < *med*, truePRE(n) = low denoted as 0
   If *mpg(n)* >= *med*, truePRE(n) = high denoted as 1
   Analysis: All cases in LOWmpg will have true class = low because the LOWmpg table gathers all the cases for which mpg <= quantile Q33%, which is always below the median. All cases in HIGHmpg will have true class = high because the HIGHmpg table gathers all the cases for which mpg > quantile Q66%, which is always above the median.

```
> truePRE = rep(0, length(Car$mpg))
> truePRE[newCar$mpg >= median(newCar$mpg)] = 1
> truePRE
  [1] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 1 1 1 1 1 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1
 [58] 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 1
[115] 0 0 1 1 0 0 0 1 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 1 1
[172] 1 0 1 0 1 1 0 1 1 1 1 1 1 1 0 0 0 0 0 0 1 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 0 0
0 0 0 1 1 1 1 1 0 0 0 0 0 0 0 0 0
[229] 0 0 0 1 1 1 1 1 1 1 1 0 0 0 1 1 1 1 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 1 1 1
0 1 1 1 0 0 0 0 1 1 0 0 0 0 0 0 0
[286] 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[343] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 0 1 1 1 1 1 1 1 1
```

4. We define the train.set as the union of { LOWmpg & HIGHpmg }. Each one of these two classes from question 8 essentially contains 1/3 of all cases.

```
> train.set <- subset(newCar1, newCar1$mpg<= quantile(newCar1$mpg, probs = c(0.33))|newCar1$mpg > quantile(newCar1$mpg, probs = c(0.66)))
> dim(train.set)
[1] 262  14
```

For each case n in this train.set, we predict its class by using the classifier Pred_A1, then compute its 2x2 confusion matrix on the training.set to evaluate the performance of Pred_A1 on the train.set.

Confusion Matrix for training set with A = 1:

```
> confusion_matrix1 = table(train.set$Pred_A1, train.set$truePRE)
> confusion_matrix1

                    truePRE
                    low high
                      0    1
  Pred_A1 low      0 129    4
          high     1   1  128
> accuracy1 = sum(diag(confusion_matrix1))/sum(confusion_matrix1)
> accuracy1
[1] 0.980916
> global accuracy1
                    truePRE
                    low     high
                    0       1
  Pred_A1 low      0 96.99%  3%
          high     1 0.78%   99.22%
```

Repeat the preceding operations for A= 0 and for A= 2, we obtain:

Confusion Matrix for training set with A = 0:

```
> confusion_matrix0 = table(train.set$Pred_A0, train.set$truePRE)
> confusion_matrix0

                    truePRE
                    low high
                      0    1
  Pred_A0 low      0 129    4
          high     1   1  128
> accuracy0
[1] 0.980916
```

```
> global accuracy0
                    truePRE
                    low     high
                    0       1
  Pred_A0 low     0 96.99%  3%
          high    1 0.78%   99.22%
```

Confusion Matrix for training set with A = 2:

```
> confusion_matrix2 = table(train.set$Pred_A2, train.set$truePRE)
> confusion_matrix2

                    truePRE
                    low high
                      0   1
  Pred_A2 low     0 129  10
          high    1   1 122
> accuracy0
[1] 0.9580153
> global accuracy2
                    truePRE
                    low     high
                    0       1
  Pred_A2 low     0 92.81%  7.19%
          high    1 0.81%   99.19%
```

5. We choose the remaining 1/3 cases which are not in the train.set will constitute the test.set.

```
> test.set <- a[ !(a$mpg %in% b$mpg), ]
> dim(test.set)
[1] 130  14
```

Applying the classifier Pred_A defined by score.full(n). for each case "n" in this test.set and computing the confusion matrix of the classifier Pred_A1, we obtain:

Confusion Matrix for testing set with A = 1:

```
> confusion_matrix1 = table(test.set$Pred_A1, test.set$truePRE)
> confusion_matrix1

                    truePRE
                    low high
                      0   1
  Pred_A1 low     0  44   9
          high    1  22  55
> accuracy1 = sum(diag(confusion_matrix1))/sum(confusion_matrix1)
> accuracy1
[1] 0.7615
```

```
> global accuracy1
                    truePRE
                    low     high
                    0       1
  Pred_A1 low     0 83.02%  16.98%
          high    1 28.57%  71.42%
```

Repeat the preceding operations for A= 0 and for A= 2, we obtain:

Confusion Matrix for testing set with A = 0:

```
> confusion_matrix0 = table(test.set$Pred_A0, test.set$truePRE)
> confusion_matrix0

                    truePRE
                    low high
                      0    1
  Pred_A0 low     0  44    9
          high    1  22   55
> accuracy0
[1] 0.7615
> global accuracy1
                    truePRE
                    low     high
                    0       1
  Pred_A0 low     0 83.02%  16.98%
          high    1 28.57%  71.42%
```

Confusion Matrix for testing set with A = 2:

```
> confusion_matrix2 = table(test.set$Pred_A2, test.set$truePRE)
> confusion_matrix2

                    truePRE
                    low high
                      0    1
  Pred_A2 low     0  43    16
          high    1  21    50
> accuracy2
[1] 0.7154
> global accuracy2
                    truePRE
                    low     high
                    0       1
  Pred_A2 low     0 72.88%  27.12%
          high    1 29.58%  70.42%
```

6. Analysis:

From the above result, we can see that for the training set, the accuracy with A = 0 and A = 1 is roughly 98%. Specifically, only 3% of the training set is predicted to be low mpg but are truly high mpg and less than 1% predicted to be high mpg but are truly low mpg. When we change to A = 2, the accuracy is lowered to 96% for the training set, with 7% falsely predicted to be low mpg and less than 1% falsely predicted to be high mpg.

For the testing set, the accuracy is much lower than the accuracy for the training set. The classifier's accuracy for A = 0 and A = 1 on the test set was 76% with 29% wrongly predicted to be low mpg and 17% wrongly predicted to be high mpg. For A = 2, the classifier's accuracy on the test set was even worse, 72%, with 30% falsely predicted to be low mpg and 27% falsely predicted to be high mpg.

As a conclusion, A = 0 or A = 1 in the preceding 3 thresholds A = 0 , 1, 2 provides the best classifier Pred_A. Even if we change to a higher threshold A = 3 or a higher number, it will not improve the accuracy, but reduce it. Our suggested improvements are we could try different thresholds for our classifier, or take out features that are not helpful in classifying between high mpg and low mpg, or even try different thresholds for our classifier. We would have to try each of the methods to evaluate which ones provide the better results, or we could combine them together.