Test Report

Jiawen Qi

March 20, 2017

1 Conceptual Question

a

Whether the appropriate method would be classification or regression?
We should use regression method, because 'Salary' is a continuous variable.

Whether we are most interested in inference or prediction?
We are most interested in inference, because we are trying to explain and interpret observations based on the evidence.

Indidate what n and p are for each section
Response variable: salary of the CEO
Predictor variables: industry, number of employees, and total profit

b

Whether the appropriate method would be classification or regression?
It's a classification problem, because there are two class of the product: success or failure.

Whether we are most interested in inference or prediction?
Prediction, because we are trying to guess about a future event.

Indidate what n and p are for each section
Response variable: whether or not the product succeeded or failed
Predictor variables: price of the product, competition price, marketing budget, ten other variables

c

Whether the appropriate method would be classification or regression?
Regression, because '% change in the dollar' is continuous not categorical.

Whether we are most interested in inference or prediction?
Prediction, because we are trying to predict the future '% change in the dollar', guess about a future event.

Indidate what n and p are for each section
Response variable: % change in the dollar

Predictor variables: % change in the market in the United States, the % change in the market in China, and the % change in the market in France.

2 Applied Question

```
cars_mileage <- read.csv("Cars_mileage.csv", header = TRUE, stringsAsFactors
= FALSE)  # import tha dataset
head(cars_mileage)  # look at the first 6 lines to have a taste

##   mpg cylinders displacement horsepower weight acceleration year origin
## 1  18         8          307        130   3504         12.0   70      1
## 2  15         8          350        165   3693         11.5   70      1
## 3  18         8          318        150   3436         11.0   70      1
## 4  16         8          304        150   3433         12.0   70      1
## 5  17         8          302        140   3449         10.5   70      1
## 6  15         8          429        198   4341         10.0   70      1
##                       name
## 1 chevrolet chevelle malibu
## 2         buick skylark 320
## 3        plymouth satellite
## 4             amc rebel sst
## 5               ford torino
## 6          ford galaxie 500

summary(cars_mileage)  # have an overview

##       mpg          cylinders      displacement    horsepower
##  Min.   : 9.00   Min.   :3.000   Min.   : 68.0   Length:397
##  1st Qu.:17.50   1st Qu.:4.000   1st Qu.:104.0   Class :character
##  Median :23.00   Median :4.000   Median :146.0   Mode  :character
##  Mean   :23.52   Mean   :5.458   Mean   :193.5
##  3rd Qu.:29.00   3rd Qu.:8.000   3rd Qu.:262.0
##  Max.   :46.60   Max.   :8.000   Max.   :455.0
##      weight      acceleration        year           origin
##  Min.   :1613   Min.   : 8.00   Min.   :70.00   Min.   :1.000
##  1st Qu.:2223   1st Qu.:13.80   1st Qu.:73.00   1st Qu.:1.000
##  Median :2800   Median :15.50   Median :76.00   Median :1.000
##  Mean   :2970   Mean   :15.56   Mean   :75.99   Mean   :1.574
##  3rd Qu.:3609   3rd Qu.:17.10   3rd Qu.:79.00   3rd Qu.:2.000
##  Max.   :5140   Max.   :24.80   Max.   :82.00   Max.   :3.000
##      name
##  Length:397
##  Class :character
##  Mode  :character
##
##
##
```

a. Create a binary variable mpg_binary

```
cars_mileage$mpg_binary <- 0
cars_mileage$mpg_binary[cars_mileage$mpg > median(cars_mileage$mpg)] <- 1
cars_mileage$mpg_binary <- as.factor(cars_mileage$mpg_binary)
```

b. Which of the other variables seem most likely to be useful in predicting whether a car's mpg is above or below its median?

Before deciding which variable seems most likely to be useful, we need to dig into each variable to understand it:

Dig into each variable

```
##### cylinders #####
summary(cars_mileage$cylinders)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   3.000   4.000   4.000   5.458   8.000   8.000

table(cars_mileage$cylinders)

##
##   3   4   5   6   8
##   4 203   3  84 103

cars_mileage$cylinders <- as.factor(cars_mileage$cylinders)
```

`cylinders` variable is the number of cylinders in a car. In this dataset, most of the cars have 4 cylinders. I would like to treat this attribute as a factor, because, you can not seperate a cynlinder, eg. 2.5 cylinders.

```
##### displacement #####
summary(cars_mileage$displacement)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    68.0   104.0   146.0   193.5   262.0   455.0

table(cars_mileage$displacement)

##
##   68   70   71   72   76   78   79   80   81   83   85   86   88   89   90
##    1    3    2    1    1    1    6    1    1    1    8    4    1    5    8
##   91   96   97 97.5   98  100  101  104  105  107  108  110  111  112  113
##   12    1   21    1   18    1    1    1    7    5    5    1    1    4    3
##  114  115  116  119  120  121  122  130  131  134  135  140  141  144  145
##    1    1    4    6    9   11    7    1    1    4    5   16    2    1    1
##  146  151  155  156  163  168  171  173  181  183  198  199  200  225  231
##    3    9    1    6    2    3    1    3    1    1    3    2    8   13    8
##  232  250  258  260  262  267  302  304  305  307  318  340  350  351  360
##   11   17    5    3    2    1   11    7    4    3   17    1   18    8    4
```

```
##  383  390  400  429  440  454  455
##    2    1   13    3    2    1    3
```

displacementvariable ranges from 68 to 455. It's a continuous variable.

```
##### horsepower #####
summary(cars_mileage$horsepower)

##    Length     Class      Mode
##       397 character character

table(cars_mileage$horsepower)

##
##   ? 100 102 103 105 107 108 110 112 113 115 116 120 122 125 129 130 132
##   5  17   1   1  12   1   1  18   3   1   5   1   4   1   3   2   5   1
## 133 135 137 138 139 140 142 145 148 149 150 152 153 155 158 160 165 167
##   1   1   1   1   2   7   1   7   1   1  22   1   2   2   1   2   4   1
## 170 175 180 190 193 198 200 208 210 215 220 225 230  46  48  49  52  53
##   5   5   5   3   1   2   1   1   1   3   1   3   1   2   3   1   4   2
##  54  58  60  61  62  63  64  65  66  67  68  69  70  71  72  74  75  76
##   1   2   5   1   2   3   1  10   1  12   6   3  12   5   6   3  14   4
##  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94
##   1   6   2   7   2   1   4   6   9   5   2  19   1  20   1   6   1   1
##  95  96  97  98
##  14   3   9   2

##### Imputing missing value #####
cars_mileage$horsepower.new = cars_mileage$horsepower
cars_mileage$horsepower.new[cars_mileage$horsepower.new == "?"] <- NA
cars_mileage$horsepower.new <- as.integer(cars_mileage$horsepower.new)
library(Hmisc)

cars_mileage$imputed_horsepower <- as.integer(with(cars_mileage, impute(horse
power.new,
    mean)))
summary(cars_mileage$imputed_horsepower)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    46.0    76.0    95.0   104.5   125.0   230.0

table(cars_mileage$imputed_horsepower)

##
##  46  48  49  52  53  54  58  60  61  62  63  64  65  66  67  68  69  70
##   2   3   1   4   2   1   2   5   1   2   3   1  10   1  12   6   3  12
##  71  72  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89
##   5   6   3  14   4   1   6   2   7   2   1   4   6   9   5   2  19   1
##  90  91  92  93  94  95  96  97  98 100 102 103 104 105 107 108 110 112
##  20   1   6   1   1  14   3   9   2  17   1   1   5  12   1   1  18   3
## 113 115 116 120 122 125 129 130 132 133 135 137 138 139 140 142 145 148
##   1   5   1   4   1   3   2   5   1   1   1   1   1   2   7   1   7   1
```

```
## 149 150 152 153 155 158 160 165 167 170 175 180 190 193 198 200 208 210
##   1  22   1   2   2   1   2   4   1   5   5   5   3   1   2   1   1   1
## 215 220 225 230
##   3   1   3   1
```

Why `horsepower` variables is treated as character? Because there are 5 missing value marked as ? not `NA`. This is so tricky here. We cannot find this kind of missing value by function `is.na()`. Therefore, I imputed the missing value using `Hmisc`. After imputing, this variable rangers from 46 to 230.

```
##### weight #####
summary(cars_mileage$weight)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1613    2223    2800    2970    3609    5140
```

`weight` ranges from 1613 to 5140.

```
##### acceleration #####
summary(cars_mileage$acceleration)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    8.00   13.80   15.50   15.56   17.10   24.80
```

`acceleration` ranges from 8 to 24.8.

```
##### year #####
summary(cars_mileage$year)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   70.00   73.00   76.00   75.99   79.00   82.00

table(cars_mileage$year)

##
## 70 71 72 73 74 75 76 77 78 79 80 81 82
## 29 28 28 40 27 30 34 28 36 29 29 29 30

cars_mileage$age <- 117 - cars_mileage$year
```

`year` is treated as integer in this dataset. I would like to calculate the `age` of a car.

```
##### origin #####
summary(cars_mileage$origin)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   1.000   1.000   1.574   2.000   3.000

table(cars_mileage$origin)

##
##   1   2   3
## 248  70  79
```

```
cars_mileage$origin <- as.factor(cars_mileage$origin)
```

Actually, I don't have the description for this dataset. All I can do is trying to guess and understand each variable. Also, I don't have much interest in cars. With my guess, the `origin` here means first hand, second hand, third hand? I'm not sure, maybe other meaning, but I think it's a categotical variable.

```
##### name #####
summary(cars_mileage$name)

##    Length     Class      Mode
##       397 character character

cars_mileage$brands <- gsub(" .*$", "", cars_mileage$name)
table(cars_mileage$brands)

##
##           amc          audi           bmw         buick      cadillac
##            27             7             2            17             2
##         capri     chevroelt     chevrolet         chevy      chrysler
##             1             1            43             3             6
##        datsun         dodge          fiat          ford            hi
##            23            28             8            51             1
##         honda         maxda         mazda      mercedes mercedes-benz
##            13             2            10             1             2
##       mercury        nissan    oldsmobile          opel       peugeot
##            11             1            10             4             8
##      plymouth       pontiac       renault          saab        subaru
##            31            16             5             4             4
##        toyota       toyouta       triumph     vokswagen    volkswagen
##            25             1             1             1            15
##         volvo            vw
##             6             6

cars_mileage$brands[cars_mileage$brands == "capri"] <- "mercury"
cars_mileage$brands[cars_mileage$brands == "chevroelt"] <- "chevrolet"
cars_mileage$brands[cars_mileage$brands == "chevy"] <- "chevrolet"
cars_mileage$brands[cars_mileage$brands == "maxda"] <- "mazda"
cars_mileage$brands[cars_mileage$brands == "mercedes"] <- "mercedes-benz"
cars_mileage$brands[cars_mileage$brands == "toyouta"] <- "toyota"
cars_mileage$brands[cars_mileage$brands == "vokswagen"] <- "volkswagen"
cars_mileage$brands[cars_mileage$brands == "vw"] <- "volkswagen"
table(cars_mileage$brands)

##
##           amc          audi           bmw         buick      cadillac
##            27             7             2            17             2
##     chevrolet      chrysler        datsun         dodge          fiat
##            47             6            23            28             8
##          ford            hi         honda         mazda mercedes-benz
##            51             1            13            12             3
```

```
##       mercury         nissan    oldsmobile          opel       peugeot 
##            12              1            10             4             8 
##      plymouth       pontiac       renault          saab        subaru 
##            31             16             5             4             4 
##        toyota       triumph    volkswagen         volvo                 
##            26              1            22             6                 
```

```
cars_mileage$brands <- as.factor(cars_mileage$brands)
```

name is a character variable. I don't know the exact type or series for each car, but at least, I know the brands! We can extract the brand for each car. After aggregate the brands, there are several interesting points:

- "capri" is not the brand of a car, it belongs to brand mercury. https://en.wikipedia.org/wiki/Mercury_Capri#Capri_II

- "chevroelt" or "chevrolet": actually on the internet, I can only find the chevrolet, so it should be a typo.

- "chevy": when I searched this brand in Google, it cames up information of chevrolet.
- "hi": I cannot find this brand on the internet. So let's keep it.
- "maxda" or "mazda": after searching, I think it's a typo, should be mazda
- "mercedes"" or "mercedes-benz": actually the full name is mercedes-benz
- "toyota" or "toyouta": another typo, should be toyota
- "vokswagen" or "volkswagen": another typo, should be volkswagen
- "vw", is the abbr of volkswagen

Correlation Matrix

Now, the dataset is:

```
dataset <- cars_mileage[, c(10, 2, 3, 5, 6, 8, 12:14)]
names(dataset)
```

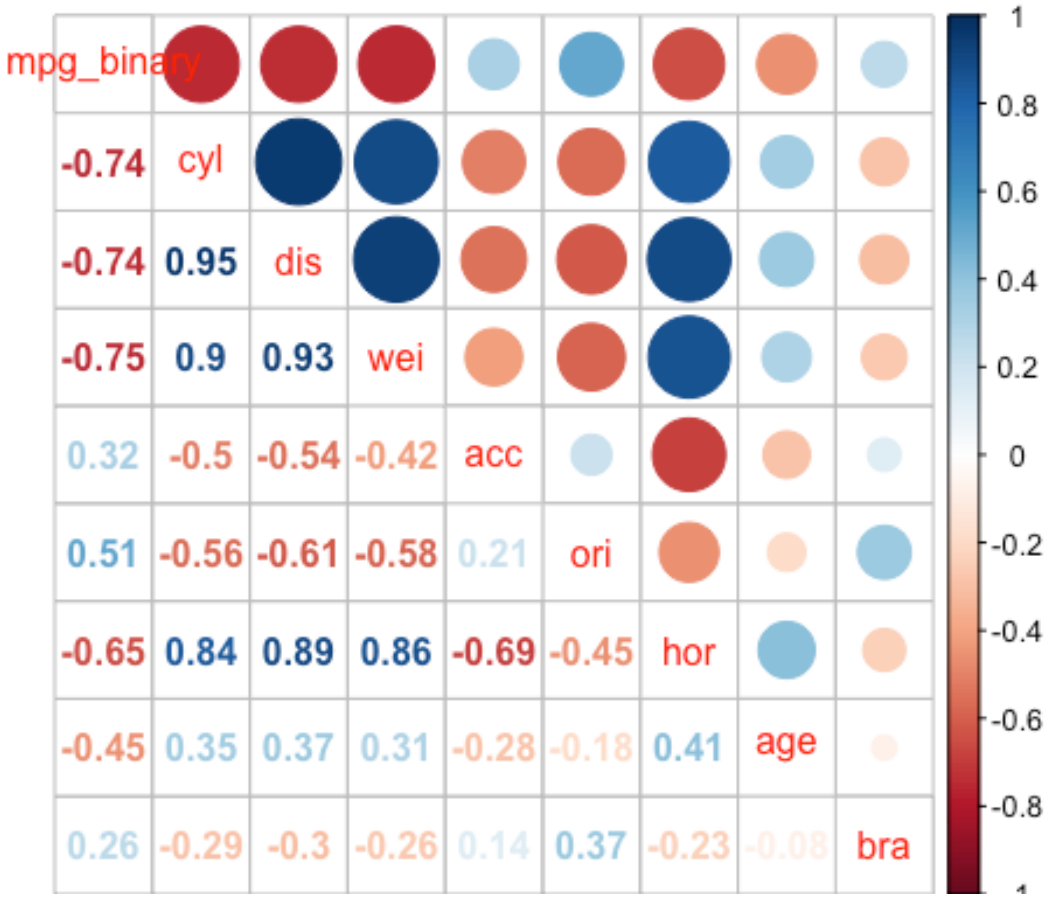
```
## [1] "mpg_binary"         "cylinders"          "displacement"
## [4] "weight"             "acceleration"       "origin"
## [7] "imputed_horsepower" "age"                "brands"
```

```
colnames(dataset) <- c("mpg_binary", "cyl", "dis", "wei", "acc", "ori", "hor"
,
    "age", "bra")
```

Make the plot:

```
library(corrplot)
dataset.num <- lapply(dataset, function(x) as.numeric(as.character(x)))  ## b
rands are changed into NA
```

```
dataset.num$bra <- as.numeric(dataset$bra)
dataset.num <- as.data.frame(dataset.num)
M <- cor(dataset.num)
corrplot.mixed(M)
```



Which variable seems most likely to be useful?

Weight it the variable seems most likely to be useful. Second, are cylinders and displacement.

c. Split the data into a training set and a test set.

I would prefer 80% training and 20% testing. (There are too many levels in brands, we will not include this feature)

```
set.seed(66666)  # my favorite seed
trainingIndex <- sample(nrow(dataset) * 0.8)
testingIndex <- setdiff(seq(1, nrow(dataset)), trainingIndex)
training <- dataset[trainingIndex, -9]
testing <- dataset[testingIndex, -9]
```

d. Perform two of the following in order to predict mpg_binary:

Logistic Regression

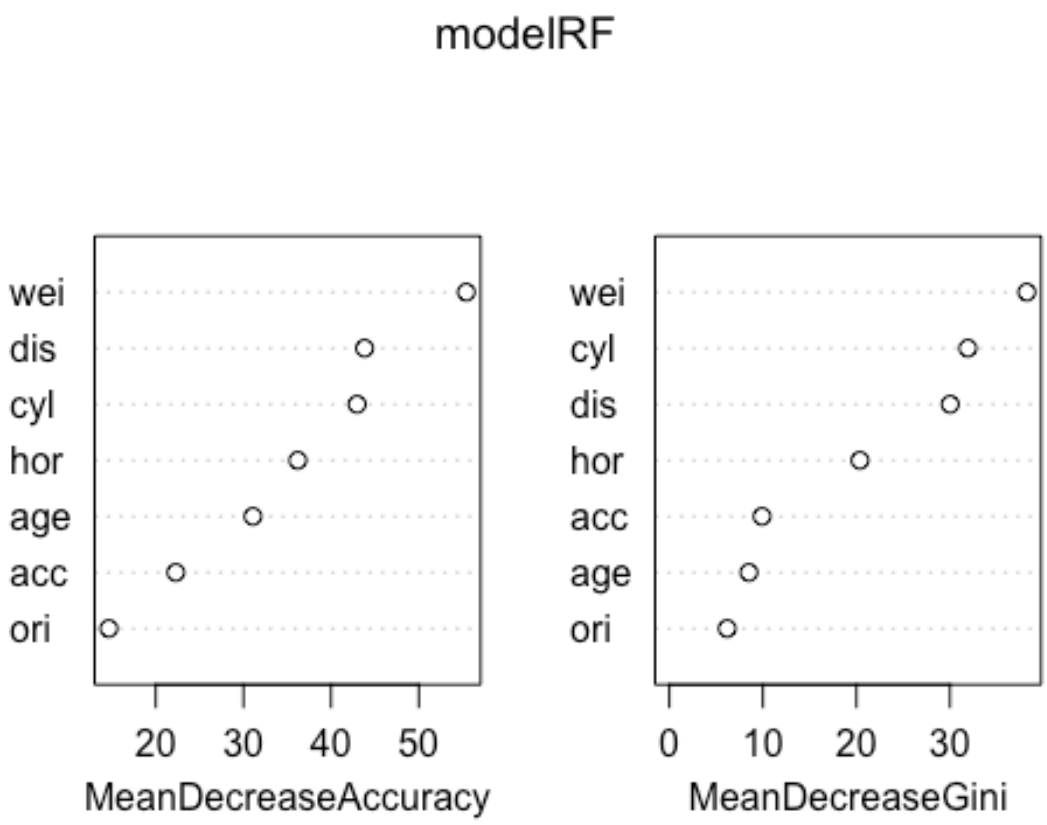
```
LGmodel <- glm(mpg_binary ~ ., family = binomial(link = "logit"), data = trai
ning)
summary(LGmodel)

##
## Call:
## glm(formula = mpg_binary ~ ., family = binomial(link = "logit"),
##     data = training)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.30672  -0.10522  -0.00961   0.13388   2.32072
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  2.026e+01  1.339e+03   0.015 0.987926
## cyl4         1.869e+01  1.339e+03   0.014 0.988863
## cyl5         1.926e+01  1.339e+03   0.014 0.988520
## cyl6         1.750e+01  1.339e+03   0.013 0.989570
## cyl8         2.079e+01  1.339e+03   0.016 0.987608
## dis         -9.010e-03  2.065e-02  -0.436 0.662681
## wei         -4.498e-03  1.669e-03  -2.695 0.007044 **
## acc         -1.422e-01  1.448e-01  -0.982 0.326344
## ori2         4.934e-01  8.646e-01   0.571 0.568204
## ori3         4.466e-01  8.267e-01   0.540 0.589031
## hor         -4.170e-02  2.708e-02  -1.540 0.123553
## age         -4.673e-01  1.211e-01  -3.859 0.000114 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 415.27  on 316  degrees of freedom
## Residual deviance: 109.97  on 305  degrees of freedom
## AIC: 133.97
##
## Number of Fisher Scoring iterations: 15

fitted.lg <- predict(LGmodel, newdata = testing, type = "response")
fitted.lg <- ifelse(fitted.lg > 0.5, 1, 0)
confusion.matrix <- table(testing$mpg_binary, fitted.lg)
accuracy <- (confusion.matrix[1, 1] + confusion.matrix[2, 2])/sum(confusion.m
atrix)
error <- 1 - accuracy
recall <- confusion.matrix[1, 1]/sum(confusion.matrix[1, ])
precision <- confusion.matrix[1, 1]/sum(confusion.matrix[, 1])
f1 <- 2 * confusion.matrix[1, 1]/(2 * confusion.matrix[1, 1] + confusion.matr
```

```
ix[1, 2] + confusion.matrix[2, 1])
paste("accuracy:", accuracy, "error:", error, "precision:", precision, "recal
l:", recall, "f1score:", f1)

## [1] "accuracy: 0.9125 error: 0.0875 precision: 0.363636363636364 recall: 1
f1score: 0.533333333333333"
```

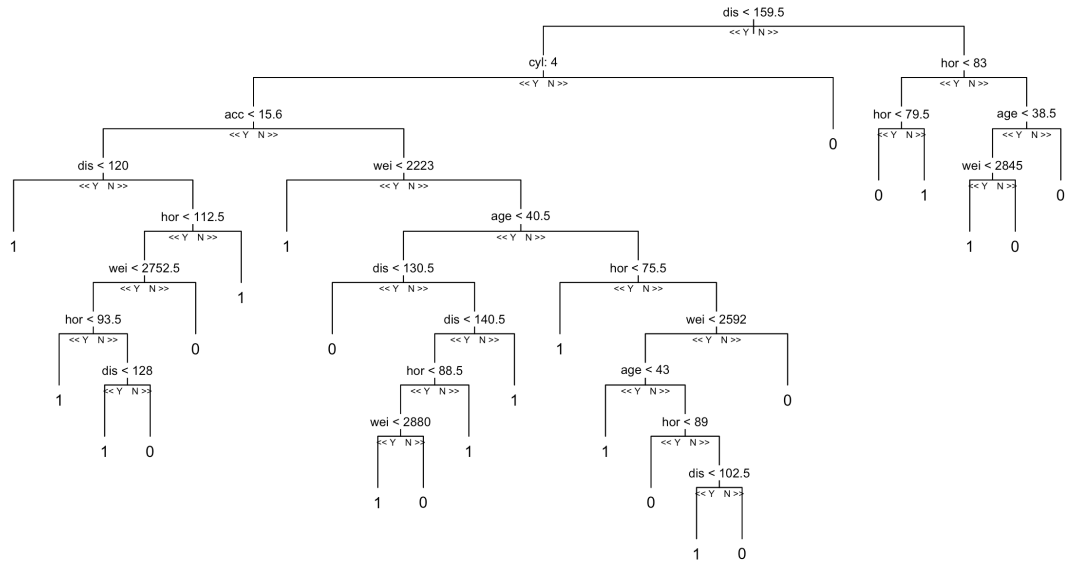
When using logistic regression to do the model, weight and age are two significant variables with negative coefficients. This is really understandable, because old and heavy car will consume more energy and the mpg will be lower, which means it's a gas guzzler. When a car is younger and lighter, it is more economical and fuel-efficient. The test error for this model is 0.0875.

Random Forest

```
library(randomForest)

library(reprtree)

set.seed(66666)
modelRF <- randomForest(mpg_binary ~ ., data = training, importance = TRUE, n
tree = 2000)
print(modelRF)

##
## Call:
##  randomForest(formula = mpg_binary ~ ., data = training, importance = TRUE
,      ntree = 2000)
##                Type of random forest: classification
##                      Number of trees: 2000
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 7.57%
## Confusion matrix:
##     0   1 class.error
## 0 189  13  0.06435644
## 1  11 104  0.09565217

varImpPlot(modelRF)
```




```
reprtree:::plot.getTree(modelRF)
```




```
prediction <- predict(modelRF, testing)
confusion.matrix <- table(testing$mpg_binary, prediction)
accuracy <- (confusion.matrix[1, 1] + confusion.matrix[2, 2])/sum(confusion.m
```

```
atrix)
error <- 1 - accuracy
recall <- confusion.matrix[1, 1]/sum(confusion.matrix[1, ])
precision <- confusion.matrix[1, 1]/sum(confusion.matrix[, 1])
f1 <- 2 * confusion.matrix[1, 1]/(2 * confusion.matrix[1, 1] + confusion.matr
ix[1, 2] + confusion.matrix[2, 1])
paste("accuracy:", accuracy, "error:", error, "precision:", precision, "recal
l:", recall, "f1score:", f1)
```

```
## [1] "accuracy: 0.8875 error: 0.1125 precision: 0.307692307692308 recall: 1
f1score: 0.470588235294118"
```

In the modeling, displacement goes at the top of the tree. The accuracy is 0.8875, and the error is 0.1125.