HW3 Problem 3

Zhengyu Lu

2022-10-18

## HW2 Prohlem 3

### (a) Create a binary variable, mpg01, that contains a 1 if mpg contains a value above its median, and a 0 if mpg contains a value below its median. You can compute the median using the median() function. Note you may find it helpful to use the data.frame() function to create a single data set containing both mpg01 and the other Auto variables.

library(ISLR)  
library(MASS)  
library(e1071)

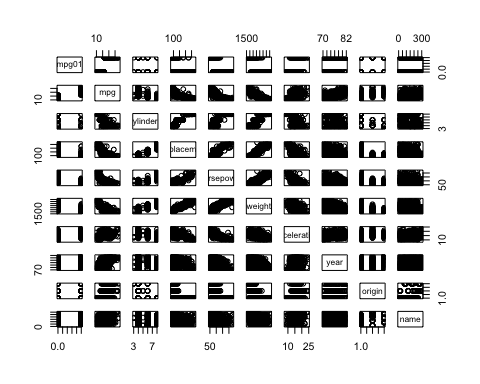
## Warning: package 'e1071' was built under R version 4.1.2

attach(Auto)  
library(class)  
mpg01 <- ifelse( mpg > median(mpg), yes = 1, no = 0)  
Auto <- data.frame(mpg01,Auto)  
head(Auto)

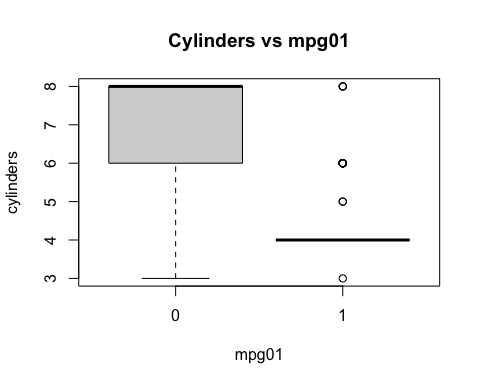
## mpg01 mpg cylinders displacement horsepower weight acceleration year origin  
## 1 0 18 8 307 130 3504 12.0 70 1  
## 2 0 15 8 350 165 3693 11.5 70 1  
## 3 0 18 8 318 150 3436 11.0 70 1  
## 4 0 16 8 304 150 3433 12.0 70 1  
## 5 0 17 8 302 140 3449 10.5 70 1  
## 6 0 15 8 429 198 4341 10.0 70 1  
## name  
## 1 chevrolet chevelle malibu  
## 2 buick skylark 320  
## 3 plymouth satellite  
## 4 amc rebel sst  
## 5 ford torino  
## 6 ford galaxie 500

### (b) Explore the data graphically in order to investigate the association between mpg01 and the other features. Which of the other features seem most likely to be useful in predicting mpg01? Scatterplots and boxplots may be useful tools to answer this question. Describe your findings.

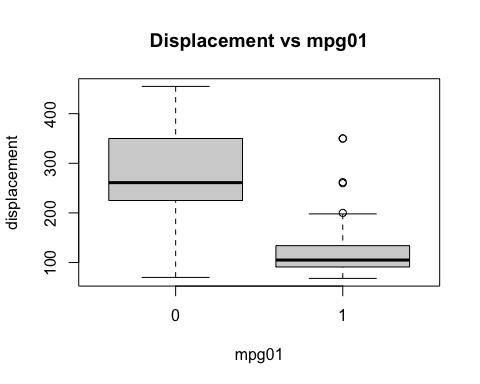
pairs(Auto)



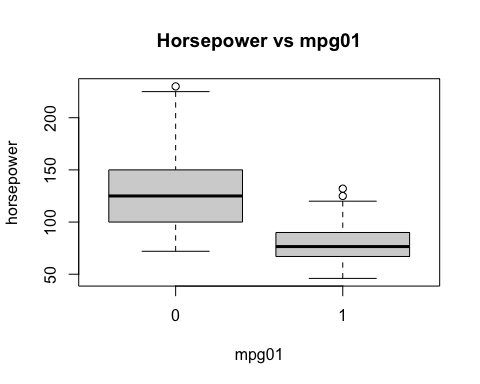
boxplot(cylinders ~ mpg01, data = Auto, main = "Cylinders vs mpg01")



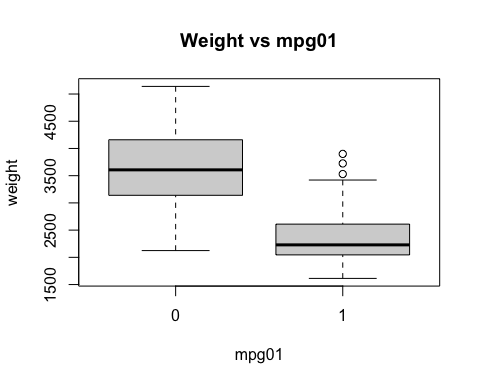
boxplot(displacement ~ mpg01, data = Auto, main = "Displacement vs mpg01")



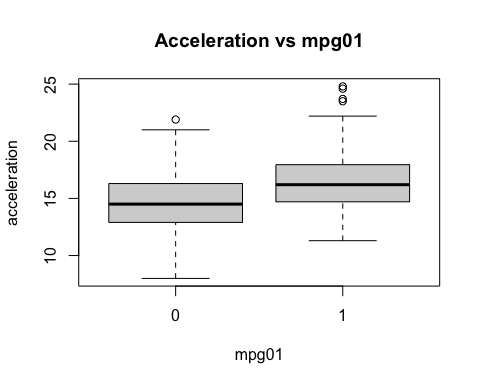
boxplot(horsepower ~ mpg01, data = Auto, main = "Horsepower vs mpg01")



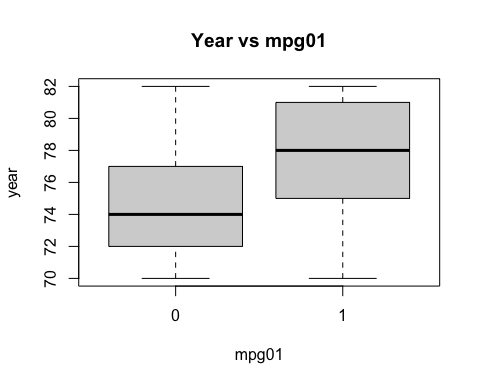
boxplot(weight ~ mpg01, data = Auto, main = "Weight vs mpg01")



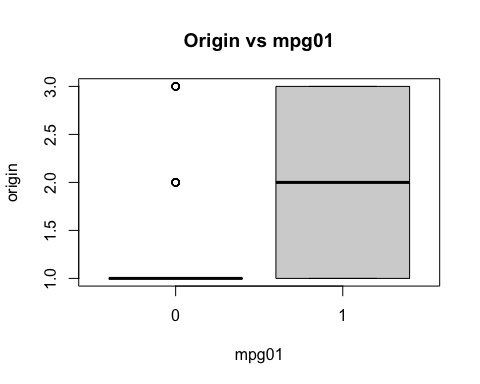
boxplot(acceleration ~ mpg01, data = Auto, main = "Acceleration vs mpg01")



boxplot(year ~ mpg01, data = Auto, main = "Year vs mpg01")



boxplot(origin ~ mpg01, data = Auto, main = "Origin vs mpg01")

 As the figures show, cylinders, weight, displacement, horsepower seem most likely to be useful in predicting mpg01. Higer cylinders will have low mpg,higher weighr will have low mpg, higher displacement will have low mpg, higher horsepower will have low mpg.

### (c) Split the data into a training set and a test set.

index <- sort(sample(nrow(Auto), nrow(Auto)\*.75))  
train <- Auto[index,]  
test <- Auto[-index,]

### (d) Perform LDA on the training data in order to predict mpg01 using the variables that seemed most associated with mpg01 in (b). What is the test error of the model obtained?

lda.model <- lda(mpg01 ~ cylinders+displacement+horsepower+weight, data = train)  
lda.predict <- predict(lda.model, test)  
table(lda.predict$class , test$mpg01)

##   
## 0 1  
## 0 43 2  
## 1 6 47

mean(lda.predict$class !=test$mpg01 )

## [1] 0.08163265

### (e) Perform QDA on the training data in order to predict mpg01 using the variables that seemed most associated with mpg01 in (b). What is the test error of the model obtained?

qda.model <- qda(mpg01 ~ cylinders+displacement+horsepower+weight, data = train)  
qda.predict <- predict(qda.model, test)  
table(qda.predict$class , test$mpg01)

##   
## 0 1  
## 0 44 5  
## 1 5 44

mean(qda.predict$class !=test$mpg01 )

## [1] 0.1020408

### (f) Perform logistic regression on the training data in order to predict mpg01 using the variables that seemed most associated with mpg01 in (b). What is the test error of the model obtained?

glm.model <- glm(mpg01 ~ cylinders+displacement+horsepower+weight, data = train)  
glm.prob <- predict(glm.model, test)  
glm.predict <- rep(0, length(glm.prob))  
glm.predict[glm.prob > .5] <- 1  
glm.predict

## [1] 0 0 1 0 1 0 1 0 0 0 0 1 1 1 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 1 0 0 1 1 0 0 1  
## [39] 1 1 1 0 0 0 0 0 0 1 1 1 1 0 0 0 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 1 1 0 0 1 0  
## [77] 0 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 1 1 1 1 1

table(glm.predict,test$mpg01)

##   
## glm.predict 0 1  
## 0 43 2  
## 1 6 47

mean(glm.predict !=test$mpg01 )

## [1] 0.08163265

### (g) Perform naive Bayes on the training data in order to predict mpg01 using the variables that seemed most associated with mpg01 in (b). What is the test error of the model obtained?

nb.fit <- naiveBayes(mpg01 ~ cylinders+displacement+horsepower+weight, data = train)  
nb.predict <- predict(nb.fit, test)  
table(nb.predict,test$mpg01)

##   
## nb.predict 0 1  
## 0 43 4  
## 1 6 45

mean(nb.predict!=test$mpg01 )

## [1] 0.1020408

### (h) Perform KNN on the training data, with several values of K, in order to predict mpg01. Use only the variables that seemed most associated with mpg01 in (b). What test errors do you obtain? Which value of K seems to perform the best on this data set?

train.X <-train[,c(3,4,5,6)]  
test.X <- test[,c(3,4,5,6)]  
train.mpg01 <-train$mpg01  
knn.predict <- knn(train.X, test.X, train.mpg01, k = 1)  
mean(knn.predict!=test$mpg01 )

## [1] 0.1122449

testError <- rep(NA, 100)  
for (k in 1:100){  
 knn.pred <- knn(train.X, test.X, train.mpg01, k = k)  
 testError[k] <- mean(knn.pred!=test$mpg01)  
}  
testError

## [1] 0.11224490 0.12244898 0.08163265 0.11224490 0.09183673 0.09183673  
## [7] 0.10204082 0.10204082 0.10204082 0.08163265 0.09183673 0.10204082  
## [13] 0.09183673 0.09183673 0.10204082 0.10204082 0.09183673 0.09183673  
## [19] 0.09183673 0.09183673 0.09183673 0.09183673 0.09183673 0.08163265  
## [25] 0.08163265 0.09183673 0.09183673 0.09183673 0.08163265 0.08163265  
## [31] 0.09183673 0.09183673 0.09183673 0.09183673 0.09183673 0.09183673  
## [37] 0.10204082 0.10204082 0.10204082 0.10204082 0.10204082 0.10204082  
## [43] 0.10204082 0.10204082 0.10204082 0.10204082 0.10204082 0.10204082  
## [49] 0.10204082 0.10204082 0.10204082 0.10204082 0.10204082 0.10204082  
## [55] 0.10204082 0.10204082 0.10204082 0.10204082 0.10204082 0.10204082  
## [61] 0.10204082 0.10204082 0.10204082 0.10204082 0.10204082 0.10204082  
## [67] 0.10204082 0.10204082 0.10204082 0.10204082 0.10204082 0.10204082  
## [73] 0.10204082 0.10204082 0.10204082 0.10204082 0.10204082 0.10204082  
## [79] 0.10204082 0.10204082 0.10204082 0.10204082 0.10204082 0.10204082  
## [85] 0.10204082 0.10204082 0.10204082 0.10204082 0.10204082 0.10204082  
## [91] 0.10204082 0.10204082 0.10204082 0.10204082 0.10204082 0.10204082  
## [97] 0.10204082 0.10204082 0.10204082 0.10204082

which.min(testError)

## [1] 3