homework4

Soumith

17/11/2021

rm(list=ls()) ### clean environment  
library(class) ###installing packages  
library(leaps)

## Warning: package 'leaps' was built under R version 4.1.1

library(ISLR2)

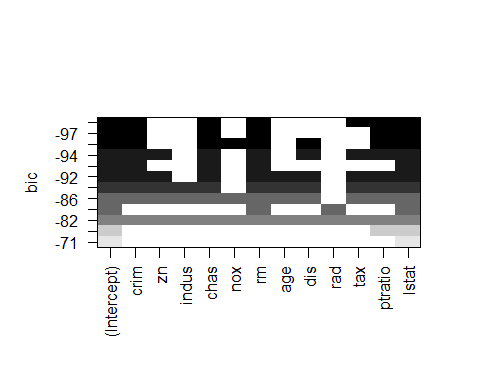
## Warning: package 'ISLR2' was built under R version 4.1.1

data(Boston) ### Dataset  
head(Boston)

## crim zn indus chas nox rm age dis rad tax ptratio lstat medv  
## 1 0.00632 18 2.31 0 0.538 6.575 65.2 4.0900 1 296 15.3 4.98 24.0  
## 2 0.02731 0 7.07 0 0.469 6.421 78.9 4.9671 2 242 17.8 9.14 21.6  
## 3 0.02729 0 7.07 0 0.469 7.185 61.1 4.9671 2 242 17.8 4.03 34.7  
## 4 0.03237 0 2.18 0 0.458 6.998 45.8 6.0622 3 222 18.7 2.94 33.4  
## 5 0.06905 0 2.18 0 0.458 7.147 54.2 6.0622 3 222 18.7 5.33 36.2  
## 6 0.02985 0 2.18 0 0.458 6.430 58.7 6.0622 3 222 18.7 5.21 28.7

set.seed(122)

indic<- sample(1:nrow(Boston),75, nrow(Boston)) ###split data into test and train data  
btrain <- Boston[indic,]  
btest <- Boston[-indic,]  
model <- regsubsets(medv~., data = btrain, nvmax = 14)  
df<- summary(model)  
plot(model)



df$bic

## [1] -70.76422 -75.52903 -85.41213 -92.66916 -96.64228 -96.83369 -95.20580  
## [8] -94.22489 -92.39732 -89.76796 -85.96928 -81.66751

df$cp

## [1] 64.063256 50.658978 32.018307 19.857878 12.977436 10.544359 10.024150  
## [8] 9.020728 8.864892 9.443454 11.012994 13.000000

df$aic

## NULL

which.min(df$cp)

## [1] 9

which.min(df$bic)

## [1] 6

predict.regsubsets = function(object, newdata, id, ...) { ### predict function  
 form = as.formula(object$call[[2]])  
 mat = model.matrix(form, newdata)  
 coefi = coef(object, id = id)  
 mat[, names(coefi)] %\*% coefi  
}   
k = 5   
set.seed(111)  
folds=sample (1:k,nrow(Boston),replace=TRUE)  
cv.errors =matrix (NA,k,14, dimnames =list(NULL , paste (1:14) ))  
  
  
for(j in 1:k){  
 best.fit=regsubsets (medv~.,data=Boston [folds!=j,],  
 nvmax=12)  
 for(i in 1:12){  
 pred=predict (best.fit ,Boston[folds ==j,],id=i)  
 cv.errors[j,i]= mean( ( Boston$medv[ folds==j]-pred)^2)  
 }  
}  
k = 10  
set.seed(111)  
folds=sample (1:k,nrow(Boston),replace=TRUE)  
cv.errors =matrix (NA,k,14, dimnames =list(NULL , paste (1:14) ))  
  
  
for(j in 1:k){  
 best.fit=regsubsets (medv~.,data=Boston [folds!=j,],  
 nvmax=12)  
 for(i in 1:12){  
 pred=predict (best.fit ,Boston[folds ==j,],id=i)  
 cv.errors[j,i]= mean( ( Boston$medv[ folds==j]-pred)^2)  
 }  
}  
library(bootstrap)

## Warning: package 'bootstrap' was built under R version 4.1.1

boot.f <- function(X\_data, Y\_data){  
 lsfit(X\_data,Y\_data)  
}  
boot.pred <- function(fit,X\_data){  
 cbind(1,X\_data)%\*%fit$coef  
}  
boot.sq.er <- function(y,yhat){  
 (y-yhat)^2  
}  
  
X\_data <- Boston[,1:12]  
Y\_data <- Boston[,13]  
  
select <- df$outmat  
err\_store <- c()  
for (i in 1:12){  
 tmp <- which(select[i,] == "\*")  
 res <- bootpred(X\_data[,tmp],Y\_data,nboot =1000,boot.f,boot.pred,boot.sq.er)  
 err\_store <- c(err\_store, res[[3]])  
}  
which.min(err\_store)

## [1] 12

2##########################

rm(list=ls()) ### clean environment  
library(tidyverse)

## Warning: package 'tidyverse' was built under R version 4.1.1

## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.5 v purrr 0.3.4  
## v tibble 3.1.5 v dplyr 1.0.7  
## v tidyr 1.1.4 v stringr 1.4.0  
## v readr 2.0.2 v forcats 0.5.1

## Warning: package 'ggplot2' was built under R version 4.1.1

## Warning: package 'tibble' was built under R version 4.1.1

## Warning: package 'tidyr' was built under R version 4.1.1

## Warning: package 'readr' was built under R version 4.1.1

## Warning: package 'purrr' was built under R version 4.1.1

## Warning: package 'dplyr' was built under R version 4.1.1

## Warning: package 'stringr' was built under R version 4.1.1

## Warning: package 'forcats' was built under R version 4.1.1

## -- Conflicts ------------------------------------------ tidyverse\_conflicts() --  
## x dplyr::filter() masks stats::filter()  
## x dplyr::lag() masks stats::lag()

library(rpart)  
library(class)  
library(corrplot)

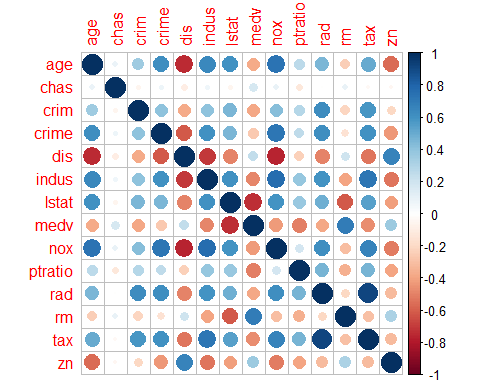
## Warning: package 'corrplot' was built under R version 4.1.1

## corrplot 0.90 loaded

data("Boston") ### viewing the boston data  
summary(Boston)

## crim zn indus chas   
## Min. : 0.00632 Min. : 0.00 Min. : 0.46 Min. :0.00000   
## 1st Qu.: 0.08205 1st Qu.: 0.00 1st Qu.: 5.19 1st Qu.:0.00000   
## Median : 0.25651 Median : 0.00 Median : 9.69 Median :0.00000   
## Mean : 3.61352 Mean : 11.36 Mean :11.14 Mean :0.06917   
## 3rd Qu.: 3.67708 3rd Qu.: 12.50 3rd Qu.:18.10 3rd Qu.:0.00000   
## Max. :88.97620 Max. :100.00 Max. :27.74 Max. :1.00000   
## nox rm age dis   
## Min. :0.3850 Min. :3.561 Min. : 2.90 Min. : 1.130   
## 1st Qu.:0.4490 1st Qu.:5.886 1st Qu.: 45.02 1st Qu.: 2.100   
## Median :0.5380 Median :6.208 Median : 77.50 Median : 3.207   
## Mean :0.5547 Mean :6.285 Mean : 68.57 Mean : 3.795   
## 3rd Qu.:0.6240 3rd Qu.:6.623 3rd Qu.: 94.08 3rd Qu.: 5.188   
## Max. :0.8710 Max. :8.780 Max. :100.00 Max. :12.127   
## rad tax ptratio lstat   
## Min. : 1.000 Min. :187.0 Min. :12.60 Min. : 1.73   
## 1st Qu.: 4.000 1st Qu.:279.0 1st Qu.:17.40 1st Qu.: 6.95   
## Median : 5.000 Median :330.0 Median :19.05 Median :11.36   
## Mean : 9.549 Mean :408.2 Mean :18.46 Mean :12.65   
## 3rd Qu.:24.000 3rd Qu.:666.0 3rd Qu.:20.20 3rd Qu.:16.95   
## Max. :24.000 Max. :711.0 Max. :22.00 Max. :37.97   
## medv   
## Min. : 5.00   
## 1st Qu.:17.02   
## Median :21.20   
## Mean :22.53   
## 3rd Qu.:25.00   
## Max. :50.00

attach(Boston) ### loading  
crime <- rep(0, length(crim)) ### creating crim variable  
crime[crim > median(crim)] <- 1   
Boston = data.frame(Boston,crime)  
train = 1:(dim(Boston)[1]/2) ### spliting the data into training set  
test = (dim(Boston)[1]/2 + 1):dim(Boston)[1] ### splitting the dataset into test set  
Boston.train = Boston[train, ]  
Boston.test = Boston[test, ]  
crime.test = crime[test]  
corrplot(cor(Boston), method="circle" ,order = "alphabet") ### Determining any associations to crimeset



set.seed(150)

Boston.fit <-glm(crime~ nox+tax+dis+rad+age+indus, data=Boston.train,family=binomial)  
Boston.probs = predict(Boston.fit, Boston.test, type = "response")   
Boston.pred = rep(0, length(Boston.probs))  
Boston.pred[Boston.probs > 0.5] = 1  
table(Boston.pred, crime.test) ### table view

## crime.test  
## Boston.pred 0 1  
## 0 75 8  
## 1 15 155

mean(Boston.pred != crime.test)

## [1] 0.09090909

summary(Boston.fit)

##   
## Call:  
## glm(formula = crime ~ nox + tax + dis + rad + age + indus, family = binomial,   
## data = Boston.train)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -1.97810 -0.21406 -0.03454 0.47107 3.04502   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -42.214032 7.617440 -5.542 2.99e-08 \*\*\*  
## nox 80.868029 16.066473 5.033 4.82e-07 \*\*\*  
## tax -0.013760 0.004956 -2.777 0.00549 \*\*   
## dis 0.307145 0.190502 1.612 0.10690   
## rad 0.847236 0.183767 4.610 4.02e-06 \*\*\*  
## age 0.003397 0.012032 0.282 0.77772   
## indus -0.213126 0.073236 -2.910 0.00361 \*\*   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 329.37 on 252 degrees of freedom  
## Residual deviance: 144.44 on 246 degrees of freedom  
## AIC: 158.44  
##   
## Number of Fisher Scoring iterations: 8

library(MASS)

## Warning: package 'MASS' was built under R version 4.1.1

##   
## Attaching package: 'MASS'

## The following object is masked \_by\_ '.GlobalEnv':  
##   
## Boston

## The following object is masked from 'package:dplyr':  
##   
## select

## The following object is masked from 'package:ISLR2':  
##   
## Boston

Boston.ldafit <-lda(crime~ indus+nox+age+dis+rad+tax, data=Boston.train,family=binomial)  
Bostonlda.pred = predict(Boston.ldafit, Boston.test)  
table(Bostonlda.pred$class, crime.test) ### table view

## crime.test  
## 0 1  
## 0 81 18  
## 1 9 145

mean(Bostonlda.pred$class != crime.test)

## [1] 0.1067194

train.K=cbind(indus,nox,age,dis,rad,tax)[train,]  
test.K=cbind(indus,nox,age,dis,rad,tax)[test,]  
Bosknn.pred=knn(train.K, test.K, crime.test, k=1)  
table(Bosknn.pred,crime.test)

## crime.test  
## Bosknn.pred 0 1  
## 0 31 155  
## 1 59 8

mean(Bosknn.pred !=crime.test)

## [1] 0.8458498

train.K=cbind(indus,nox,age,dis,rad,tax)[train,]  
test.K=cbind(indus,nox,age,dis,rad,tax)[test,]  
Bosknn.pred=knn(train.K, test.K, crime.test, k=100)  
table(Bosknn.pred,crime.test)

## crime.test  
## Bosknn.pred 0 1  
## 0 21 8  
## 1 69 155

mean(Bosknn.pred !=crime.test)

## [1] 0.3043478

library(tree)

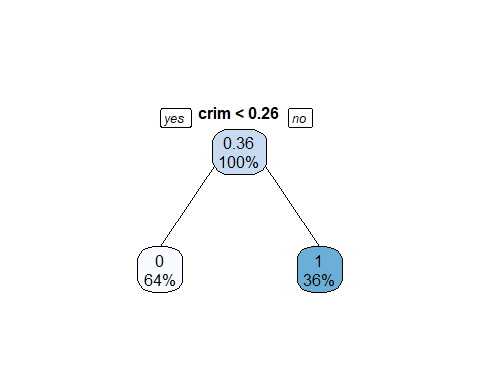
## Warning: package 'tree' was built under R version 4.1.1

## Registered S3 method overwritten by 'tree':  
## method from  
## print.tree cli

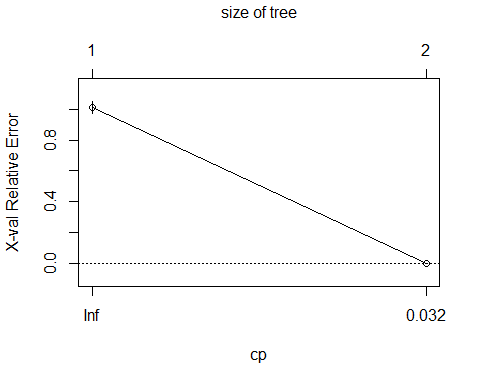
library(rpart.plot)

## Warning: package 'rpart.plot' was built under R version 4.1.1

library(rpart)  
Boston.tree <- rpart(crime ~ ., data = Boston.train, cp = 0.001)  
rpart.plot(Boston.tree, type = 1, fallen.leaves = FALSE)



plotcp(Boston.tree)



printcp(Boston.tree)

##   
## Regression tree:  
## rpart(formula = crime ~ ., data = Boston.train, cp = 0.001)  
##   
## Variables actually used in tree construction:  
## [1] crim  
##   
## Root node error: 57.984/253 = 0.22919  
##   
## n= 253   
##   
## CP nsplit rel error xerror xstd  
## 1 1.000 0 1 1.011 0.038418  
## 2 0.001 1 0 0.000 0.000000

prune.tree1 <- prune(Boston.tree, cp = 0.0085790)  
prune.tree1

## n= 253   
##   
## node), split, n, deviance, yval  
## \* denotes terminal node  
##   
## 1) root 253 57.98419 0.3557312   
## 2) crim< 0.25651 163 0.00000 0.0000000 \*  
## 3) crim>=0.25651 90 0.00000 1.0000000 \*

boston.train.pred.tree = predict(prune.tree1)  
boston.test.pred.tree = predict(prune.tree1, Boston.test)  
mean((boston.test.pred.tree - Boston.test$crime)^2)

## [1] 0

mean((boston.train.pred.tree - Boston.train$crime)^2)

## [1] 0

3a

library(ggthemes) ### calling libraries

## Warning: package 'ggthemes' was built under R version 4.1.1

attach(Auto) ### attaching data

## The following object is masked from package:ggplot2:  
##   
## mpg

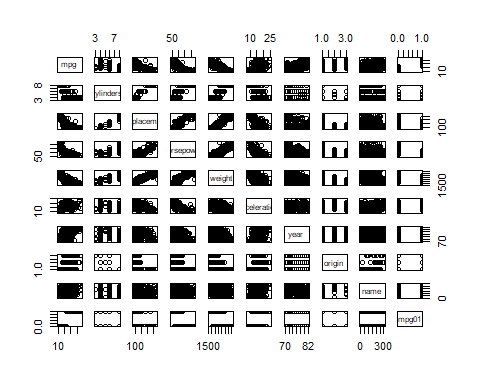
mpg01 = rep(0, length(mpg)) ### Creating binary variable mpg01 with the condition  
mpg01[mpg > median(mpg)] = 1 ### condition  
Auto = data.frame(Auto, mpg01) ### Loading the data  
head(Auto)

## 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 mpg01  
## 1 chevrolet chevelle malibu 0  
## 2 buick skylark 320 0  
## 3 plymouth satellite 0  
## 4 amc rebel sst 0  
## 5 ford torino 0  
## 6 ford galaxie 500 0

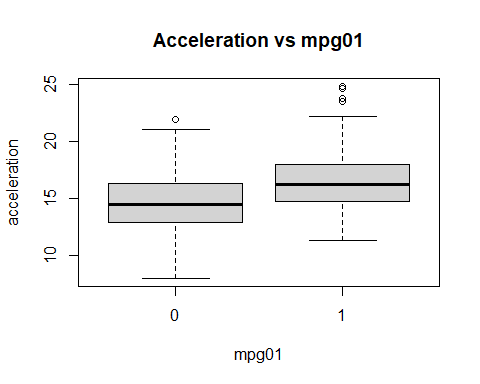
#####3b  
cor(Auto[, -9]) ### correlating

## mpg cylinders displacement horsepower weight  
## mpg 1.0000000 -0.7776175 -0.8051269 -0.7784268 -0.8322442  
## cylinders -0.7776175 1.0000000 0.9508233 0.8429834 0.8975273  
## displacement -0.8051269 0.9508233 1.0000000 0.8972570 0.9329944  
## horsepower -0.7784268 0.8429834 0.8972570 1.0000000 0.8645377  
## weight -0.8322442 0.8975273 0.9329944 0.8645377 1.0000000  
## acceleration 0.4233285 -0.5046834 -0.5438005 -0.6891955 -0.4168392  
## year 0.5805410 -0.3456474 -0.3698552 -0.4163615 -0.3091199  
## origin 0.5652088 -0.5689316 -0.6145351 -0.4551715 -0.5850054  
## mpg01 0.8369392 -0.7591939 -0.7534766 -0.6670526 -0.7577566  
## acceleration year origin mpg01  
## mpg 0.4233285 0.5805410 0.5652088 0.8369392  
## cylinders -0.5046834 -0.3456474 -0.5689316 -0.7591939  
## displacement -0.5438005 -0.3698552 -0.6145351 -0.7534766  
## horsepower -0.6891955 -0.4163615 -0.4551715 -0.6670526  
## weight -0.4168392 -0.3091199 -0.5850054 -0.7577566  
## acceleration 1.0000000 0.2903161 0.2127458 0.3468215  
## year 0.2903161 1.0000000 0.1815277 0.4299042  
## origin 0.2127458 0.1815277 1.0000000 0.5136984  
## mpg01 0.3468215 0.4299042 0.5136984 1.0000000

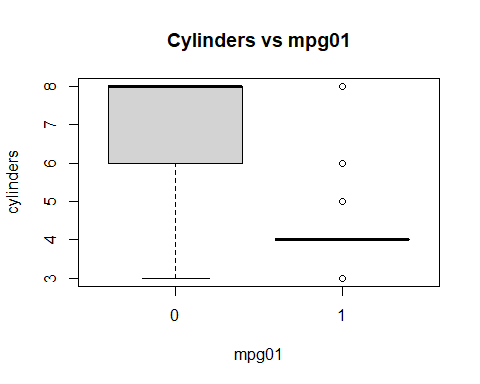
pairs(Auto)



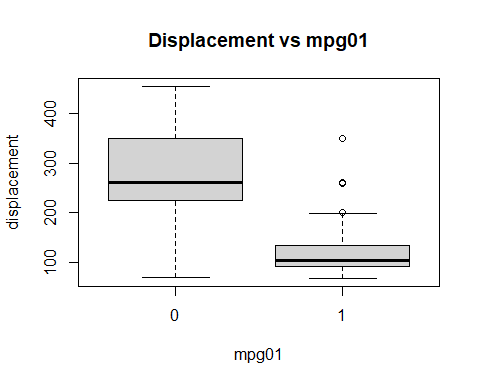
boxplot(acceleration ~ mpg01, data = Auto, main = "Acceleration vs mpg01") ### Box plots for all the variables with mpg01



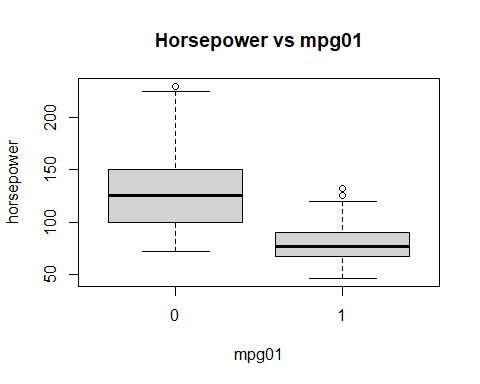
boxplot(cylinders ~ mpg01, data = Auto, main = "Cylinders vs mpg01")



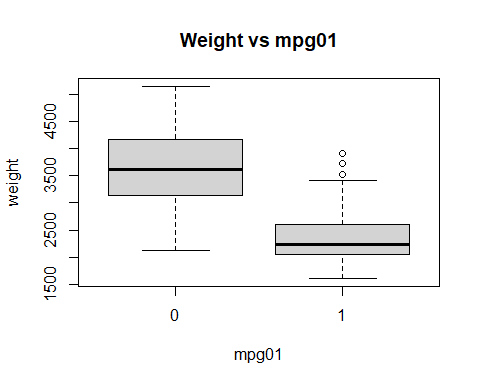
boxplot(displacement ~ mpg01, data = Auto, main = "Displacement vs mpg01")



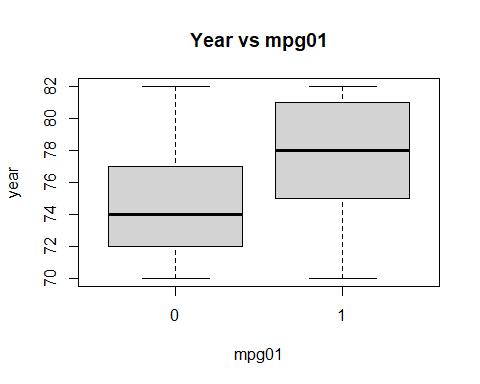
boxplot(horsepower ~ mpg01, data = Auto, main = "Horsepower vs mpg01")



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



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



####3c  
train = (year%%2 == 0) ### splitting dataset w.r.t year  
Auto.train = Auto[train, ] ### Training set  
Auto.test = Auto[!train, ] ### Test set  
mpg01.test = mpg01[!train] ### mpg01 testset

####3d  
lda.fit = lda(mpg01 ~ cylinders + displacement + horsepower + weight, data = Auto, subset = train) ### LDA on the training data in order to predict mpg01  
lda.pred = predict(lda.fit, Auto.test)   
lda.class = lda.pred$class  
table(lda.class, mpg01.test)

## mpg01.test  
## lda.class 0 1  
## 0 86 9  
## 1 14 73

mean(lda.class != mpg01.test) ### Test error 0.1263736

## [1] 0.1263736

######3e  
qda.fit = qda(mpg01 ~ cylinders + displacement + horsepower + weight, data = Auto, subset = train) ### QDA on the training data in order to predict mpg01  
qda.fit

## Call:  
## qda(mpg01 ~ cylinders + displacement + horsepower + weight, data = Auto,   
## subset = train)  
##   
## Prior probabilities of groups:  
## 0 1   
## 0.4571429 0.5428571   
##   
## Group means:  
## cylinders displacement horsepower weight  
## 0 6.812500 271.7396 133.14583 3604.823  
## 1 4.070175 111.6623 77.92105 2314.763

qda.pred = predict(qda.fit, Auto.test)  
qda.class = qda.pred$class  
table(qda.class, mpg01.test)

## mpg01.test  
## qda.class 0 1  
## 0 89 13  
## 1 11 69

mean(qda.class != mpg01.test) ### Test error 0.1318681

## [1] 0.1318681

#######3f  
glm.fit = glm(mpg01 ~ cylinders + displacement + horsepower + weight, data = Auto, subset = train, family = binomial) ### logistic regression on the training data in order to predict mpg01  
summary(glm.fit)$coef

## Estimate Std. Error z value Pr(>|z|)  
## (Intercept) 17.658730372 3.409012230 5.1800138 2.218695e-07  
## cylinders -1.028031664 0.653606999 -1.5728590 1.157515e-01  
## displacement 0.002461740 0.015029620 0.1637926 8.698944e-01  
## horsepower -0.050610857 0.025209015 -2.0076491 4.468060e-02  
## weight -0.002922352 0.001137367 -2.5694006 1.018746e-02

glm.probs = predict(glm.fit, Auto.test, type = "response")  
glm.pred = rep(0, length(glm.probs))  
glm.pred[glm.probs > .5] = 1  
table(glm.pred, mpg01.test)

## mpg01.test  
## glm.pred 0 1  
## 0 89 11  
## 1 11 71

mean(glm.pred != mpg01.test) ### Test error 0.1208791

## [1] 0.1208791

### KNN on the training data, with several values of KK, in order to predict mpg01  
  
####3g  
  
train.X = cbind(cylinders, displacement, horsepower, weight)[train, ]  
test.X = cbind(cylinders, displacement, horsepower, weight)[!train, ]  
train.mpg01 = mpg01[train]  
set.seed(1)  
knn.pred = knn(train.X, test.X, train.mpg01, k = 1) ### k=1  
table(knn.pred, mpg01.test )

## mpg01.test  
## knn.pred 0 1  
## 0 83 11  
## 1 17 71

mean(knn.pred != mpg01.test) ### Test error 0.1538462

## [1] 0.1538462

knn.pred = knn(train.X, test.X, train.mpg01, k = 10) ### k=10  
table(knn.pred, mpg01.test )

## mpg01.test  
## knn.pred 0 1  
## 0 77 7  
## 1 23 75

mean(knn.pred != mpg01.test) ### Test error 0.1648352

## [1] 0.1648352

knn.pred = knn(train.X, test.X, train.mpg01, k = 100) ### k =100  
table(knn.pred, mpg01.test )

## mpg01.test  
## knn.pred 0 1  
## 0 81 7  
## 1 19 75

mean(knn.pred != mpg01.test) ### Test error 0.1428571

## [1] 0.1428571

knn.pred = knn(train.X, test.X, train.mpg01, k = 500) ### k =500

## Warning in knn(train.X, test.X, train.mpg01, k = 500): k = 500 exceeds number  
## 210 of patterns

table(knn.pred, mpg01.test )

## mpg01.test  
## knn.pred 0 1  
## 0 0 0  
## 1 100 82

mean(knn.pred != mpg01.test) ### Test error 0.5494505

## [1] 0.5494505

### K = 100 seems to be the best KNN model with an accuracy of 0.1428571 at the rate of 14.28%