Stats 101C HW 2

Anna Piskun

10/21/2020

From textbook section 4.7:

Problem 4

When the number of features p is large, there tends to be a deterioration in the performance of KNN and other local approaches that perform prediction using only observations that are near the test observation for which a prediction must be made. This phenomenon is known as the curse of dimensionality, and it ties into the fact that non-parametric approaches often perform poorly when p is large. We will now investigate this curse.

a) Since we want only observations that are within 10% of range of X closest to that test observation (with p=1 feature), the average fraction of available observations we need in order to make a prediction is 0.1. Thus, we can conclude that we need 10% of the available observations to make a prediction.

b) Given that X1 and X2 are uniformly distributed, we can also assume independence. Thus we can find the average fraction by multiplying the two predictors together using the same formula as seen from our above calculation. If we want only responses within 10% of the range of X1 and X2, then on average we need 0.1 x 0.1= 0.01% of available observations to make our prediction.

c) Since the set of observations are again uniformly and independently distributed ranging in value from [0,1] we can use the same logic as above. If p=100 then we would just have to find the product of 0.1 x 0.1 x.... all 100 times. This gives us (0.1)ˆ100 which is approximately 0. Thus, 0% of the observations will be available to us preventing us from making a prediction.

d) We can see from a-c that as p gets larger, the number of observations actually available to us gets significantly smaller eventually reaching 0. We can expand this finding to see that as the number of p features gets infinitely large, the resulting available observations goes to 0. Here we can see the curse of dimensionality in that KNN no longer preforms properly and we are unable to perform predictions.

e) Looking at a p-dimensional hyper-cube centered around the test observation that contains, on average, 10% of the training observations, we find that for p=1 the length of each side is 0.1. For p=2, each side has a length of 0.1ˆ(1/2), and for p=100 each side has a length of 0.1ˆ(1/100).

Problem 8

Suppose that we take a data set, divide it into equally-sized training and test sets, and then try out two different classification procedures. First we use logistic regression and get an error rate of 20 % on the training data and 30 % on the test data. Next we use 1-nearest neighbors (i.e. K = 1) and get an average error rate (averaged over both test and training data sets) of 18%. Based on these results, which method should we prefer to use for classification of new observations? Why?

When we use KNN classification, with 1-nearest neighbors or when K=1, we get a training error rate of 0. That is because for any training observation its nearest neighbor will be the response itself. If we only use K=1, we are looking at 1 nearest neighbor and as the data is apart of the training data, its nearest neighbor will be the response itself. As such, if there is an average error rate of 18%, that means that the test error

rate is 36% which is higher than the 30% error rate using logistic regression. Thus, logistic regression would be the preferred method for classification of new observations due to its lower error rate.

Problem 11

In this problem, you will develop a model to predict whether a given car gets high or low gas mileage based on the Auto data set.

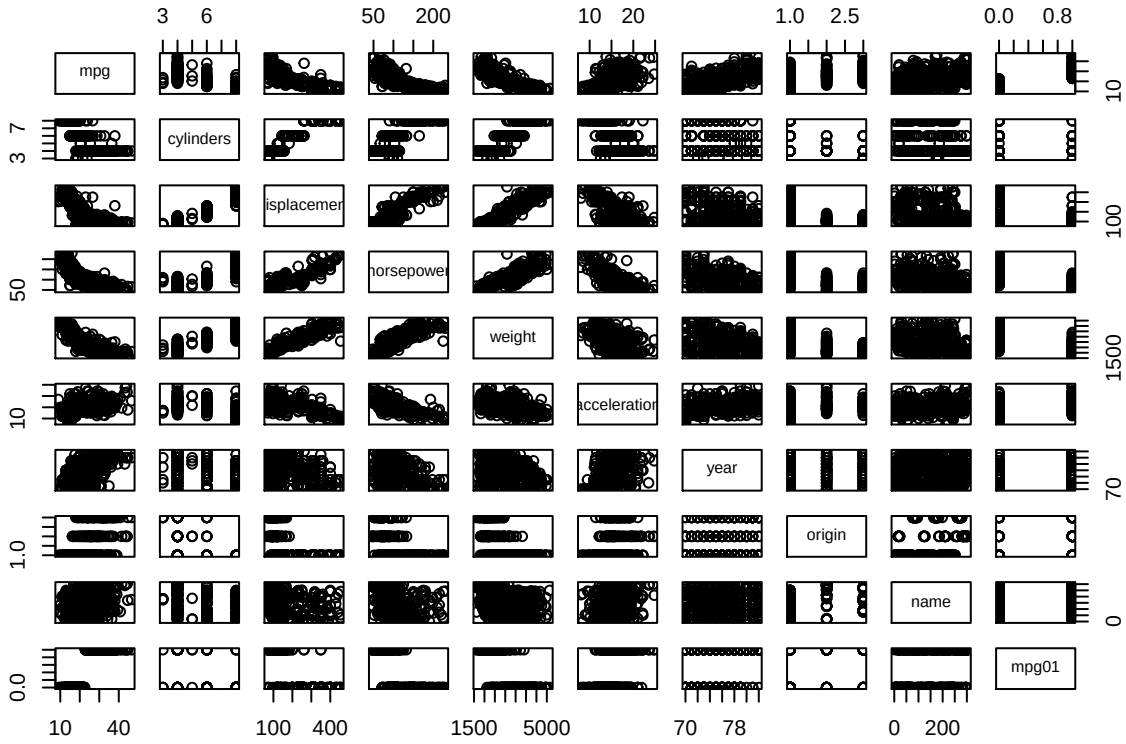
```
library(ISLR)
```

(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.

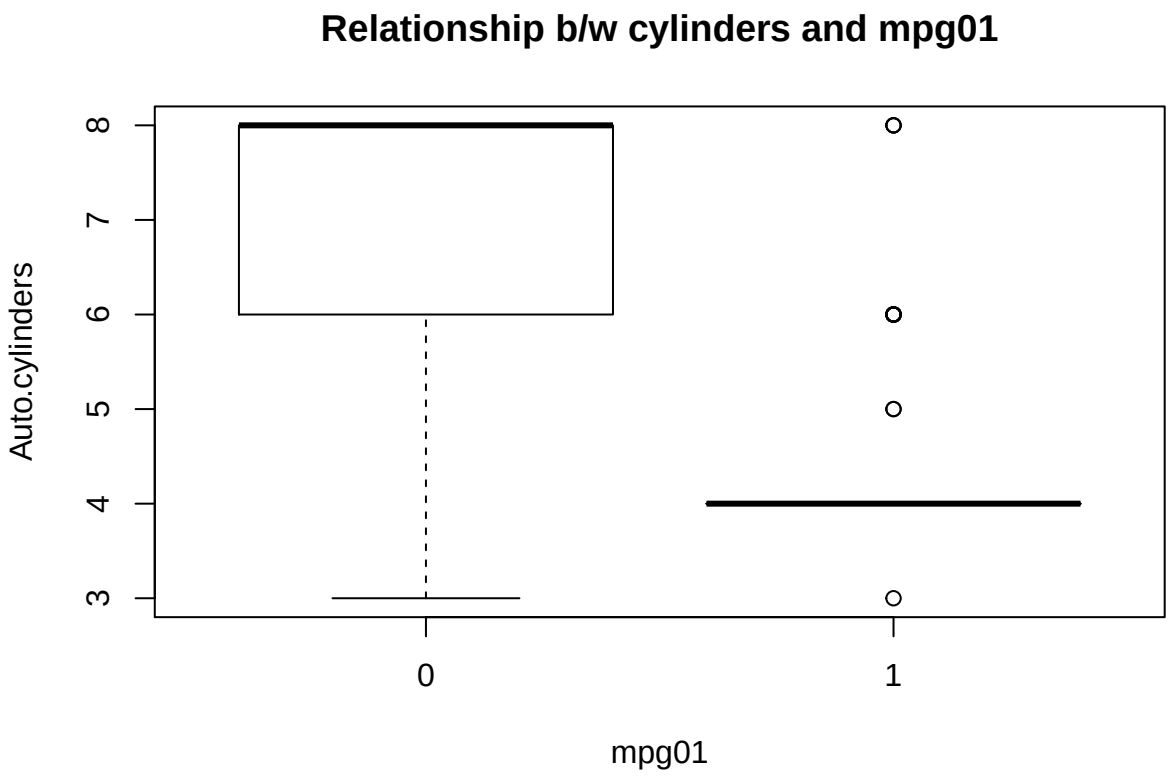
```
median <- median(Auto$mpg)

Auto$mpg01 <- ifelse(Auto$mpg > median, 1, 0)

mpg01 <- ifelse(Auto$mpg > median, 1, 0)

auto <- data.frame(mpg01, Auto$cylinders, Auto$displacement, Auto$horsepower, Auto$weight, Auto$acceler
```

(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.

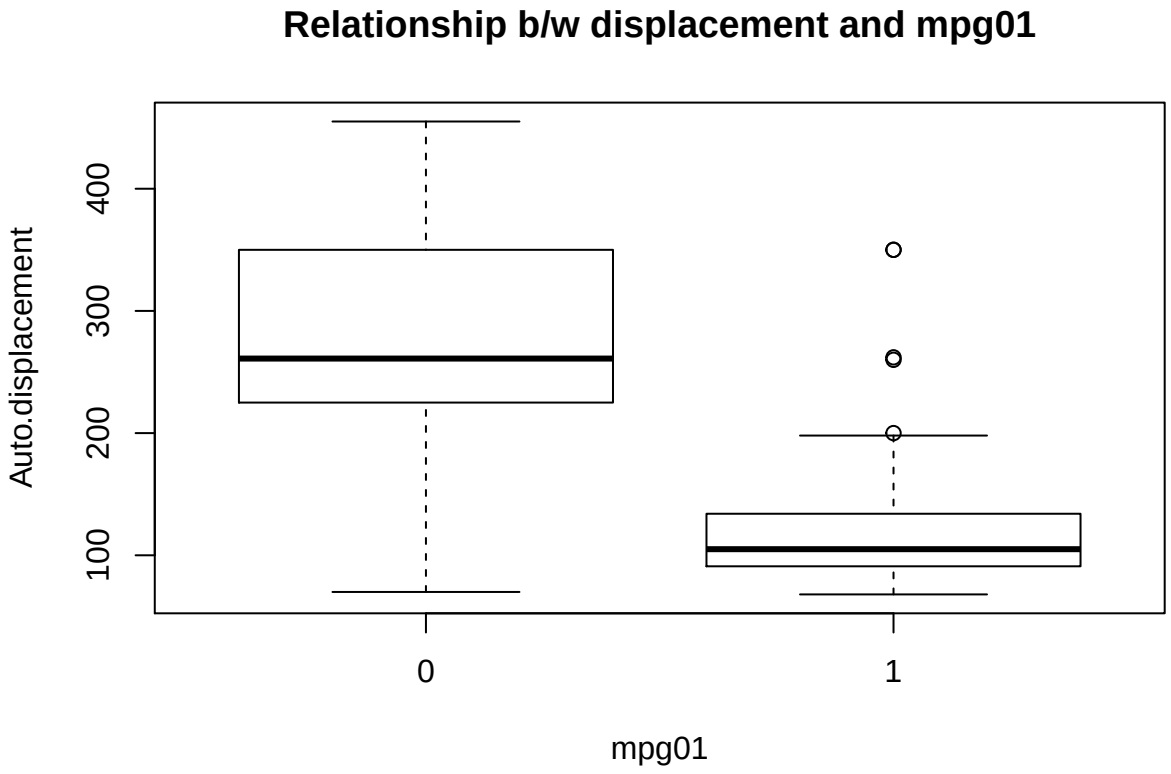
```
pairs(Auto)
```



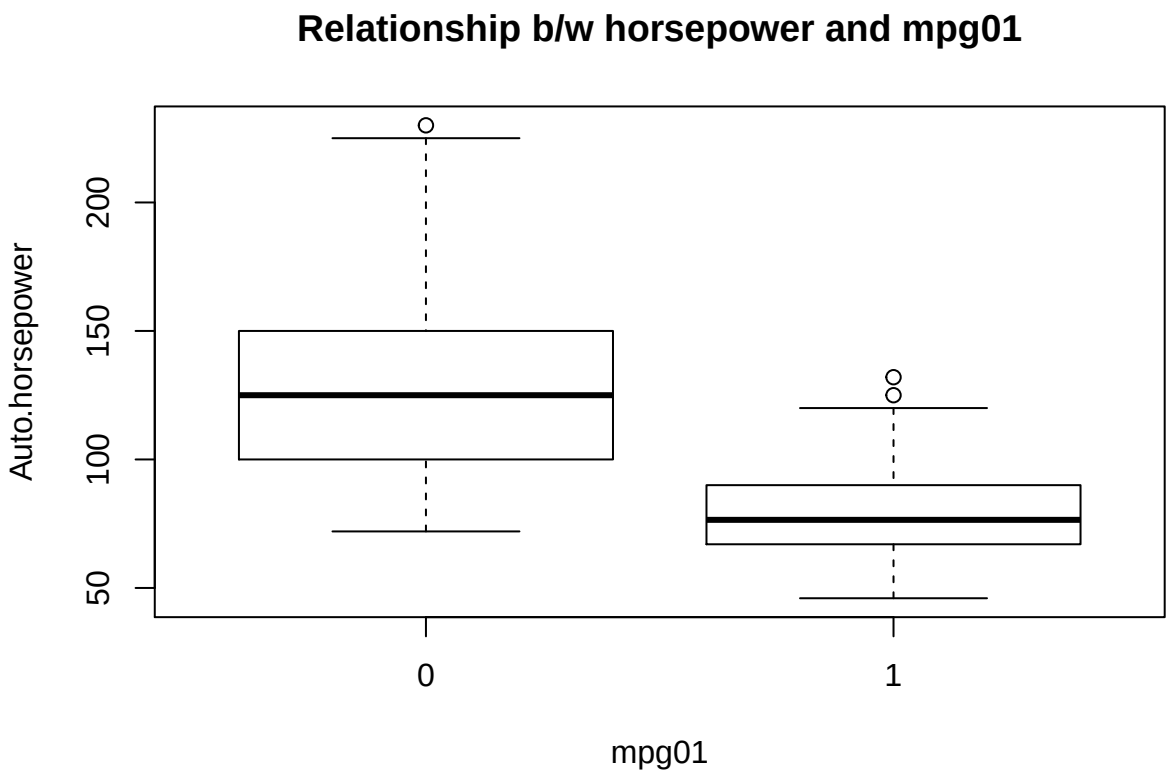

```
boxplot(Auto.cylinders~mpg01, data = auto, main = "Relationship b/w cylinders and mpg01")
```



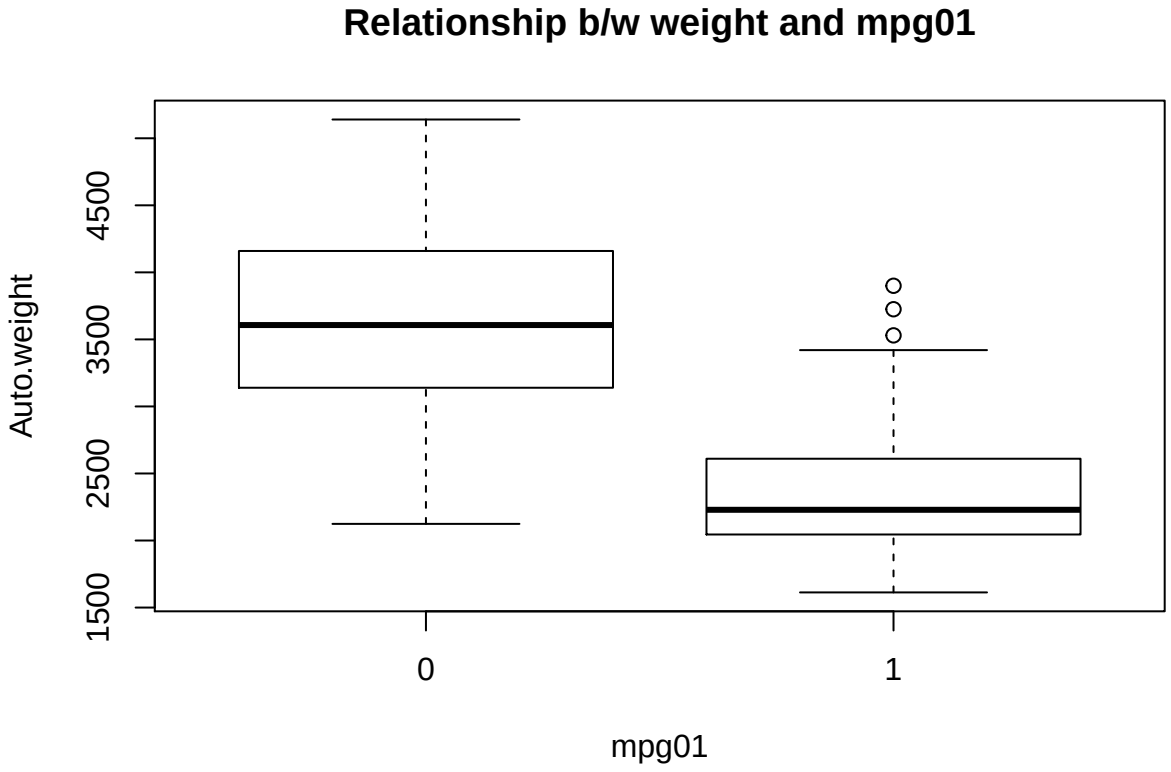

```
boxplot(Auto.displacement~mpg01, data = auto, main = "Relationship b/w displacement and mpg01")
```



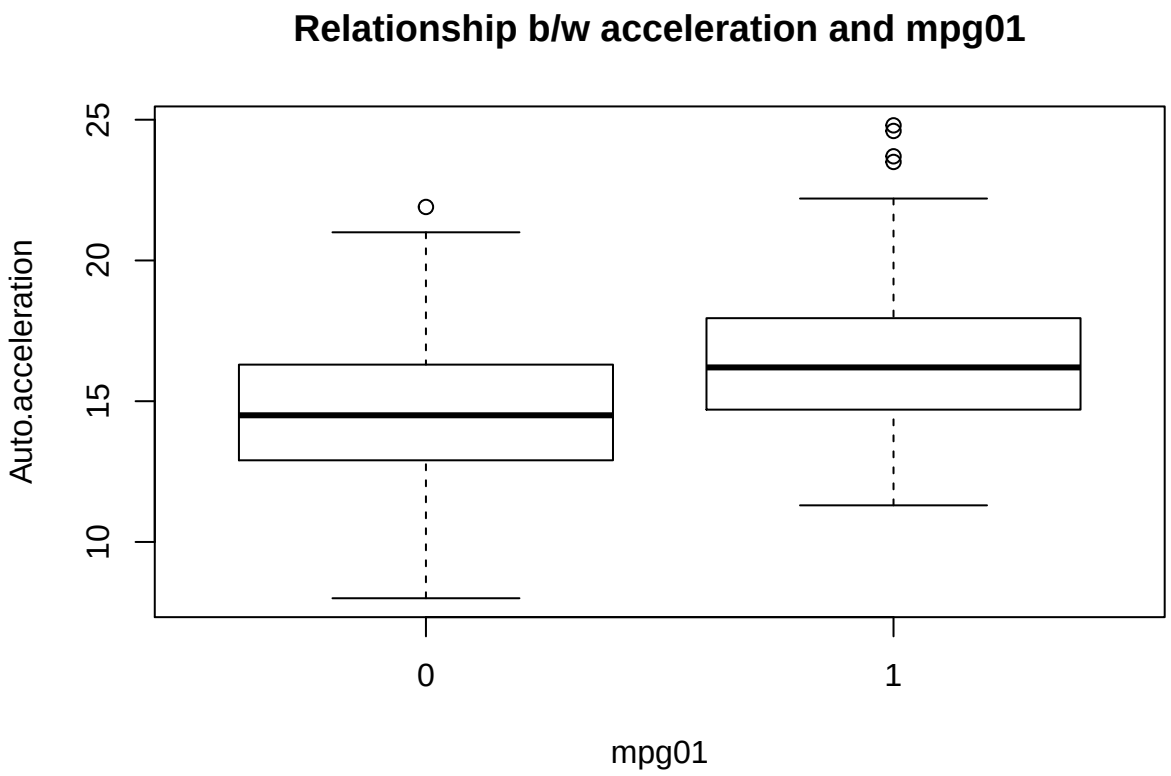

```
boxplot(Auto.horsepower~mpg01, data = auto, main = "Relationship b/w horsepower and mpg01")
```



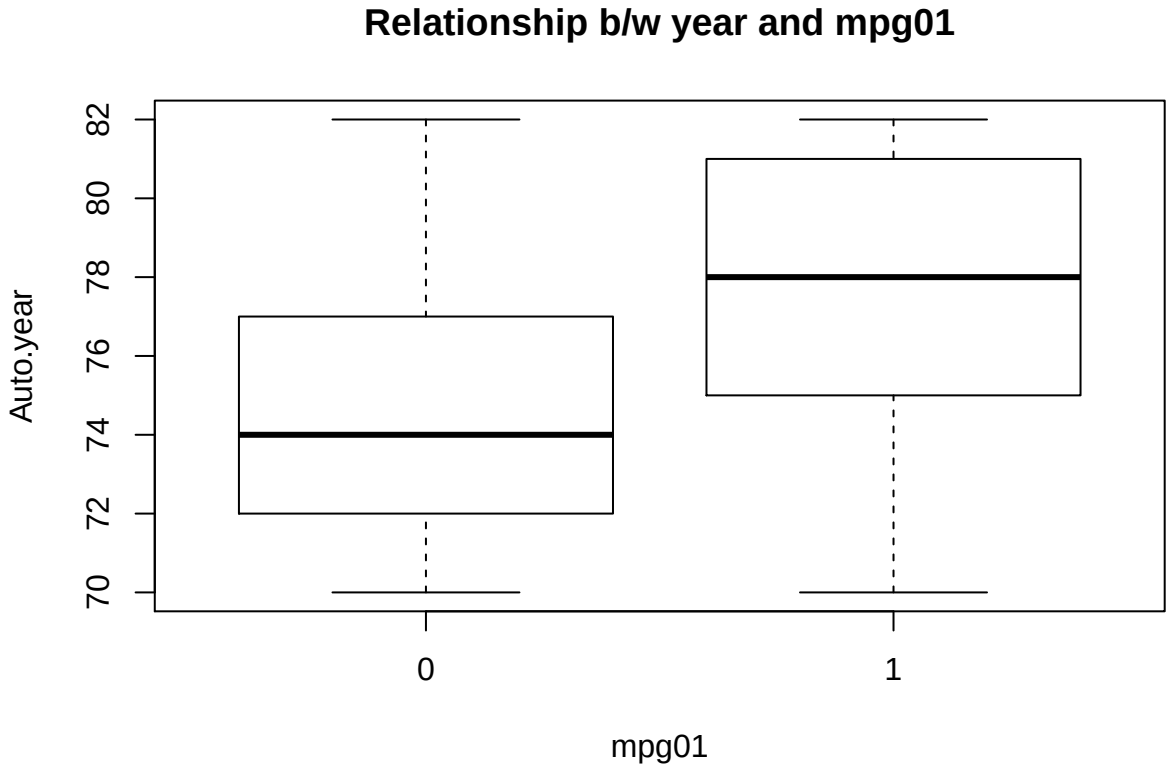

```
boxplot(Auto.weight~mpg01, data = auto, main = "Relationship b/w weight and mpg01")
```




```
boxplot(Auto.acceleration~mpg01, data = auto, main = "Relationship b/w acceleration and mpg01")
```




```
boxplot(Auto.year~mpg01, data = auto, main = "Relationship b/w year and mpg01")
```



Looking at the above scatterplots and boxplots, we can see a potential inverse relationship between *mpg*01 and *cylinders*, *displacement*, *horsepower*, *weight* and a positive relationship with *year*, which makes sense given that newer cars incorporate newer technology that may them last longer/more fuel efficient/etc.

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

```
set.seed(123456) # Set seed to reproduce results.
i <- 1:dim(auto)[1]
i.train <- sample(i, 275, replace = F) # Generate a random sample.

auto.train <- auto[i.train,]
auto.test <- auto[-i.train,]

head(auto.train)
```

```
##     mpg01 Auto.cylinders Auto.displacement Auto.horsepower Auto.weight
## 60      0              4               140              90        2408
## 234     1              4                97              75        2265
## 42      0              8               383             180        4955
## 369     1              4               140              92        2865
## 126     0              6               232             100        2901
## 199     0              6               250              78        3574
##     Auto.acceleration Auto.year Auto.origin               Auto.name
## 60               19.5        72           1          chevrolet vega
## 234              18.2        77           3 toyota corolla liftback
## 42               11.5        71           1       dodge monaco (sw)
## 369              16.4        82           1    ford fairmont futura
## 126              16.0        74           1              amc hornet
## 199              21.0        76           1       ford granada ghia
```

(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?

```
library(MASS)
lda.mod <- lda(mpg01~Auto.cylinders + Auto.displacement + Auto.horsepower + Auto.weight, data = auto.tra
lda.mod
```

```
## Call:
## lda(mpg01 ~ Auto.cylinders + Auto.displacement + Auto.horsepower +
##     Auto.weight, data = auto.train)
##
## Prior probabilities of groups:
##         0         1
## 0.4981818 0.5018182
##
## Group means:
##   Auto.cylinders Auto.displacement Auto.horsepower Auto.weight
## 0       6.605839          266.1752       126.46715    3587.453
## 1       4.210145          116.5616        79.22464    2327.130
##
## Coefficients of linear discriminants:
##                              LD1
## Auto.cylinders    -0.3075398829
## Auto.displacement -0.0009752856
## Auto.horsepower    0.0055547812
## Auto.weight       -0.0013824970
```

```
pred.lda.test <- predict(lda.mod, auto.test[,-1])

pred.lda.test$posterior[1:10,]
```

```
##             0           1
## 4  0.95566482 0.044335185
## 7  0.99595375 0.004046254
## 14 0.78597634 0.214023657
## 18 0.36992907 0.630070934
## 19 0.02286126 0.977138743
## 20 0.01525383 0.984746170
## 23 0.04605966 0.953940338
## 27 0.99593603 0.004063968
## 28 0.99557138 0.004428625
## 32 0.02959577 0.970404225
```

```
pred.lda.test$class[1:10]
```

```
##  [1] 0 0 0 1 1 1 1 0 0 1
## Levels: 0 1
```

```
# Set the standard threshold
my.thres <- 0.5
predicted.above.lda <- pred.lda.test$posterior[,"1"] > my.thres

# Confusion matrix.
table('Reference' = auto.test[,1] == '1', "Predicted" = predicted.above.lda)
```

```
##           Predicted
## Reference FALSE TRUE
##     FALSE    51    8
##     TRUE      4   54
```

```
test.error.lda <- 12/117

test.error.lda
```

```
## [1] 0.1025641
```

The test error of the LDA model obtained is approximately 10.256%.

(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?

```
library(MASS)
qda.mod <- qda(mpg01~Auto.cylinders + Auto.displacement + Auto.horsepower + Auto.weight, data = auto.tra
qda.mod
```

```
## Call:
## qda(mpg01 ~ Auto.cylinders + Auto.displacement + Auto.horsepower +
##     Auto.weight, data = auto.train)
##
## Prior probabilities of groups:
##         0         1
## 0.4981818 0.5018182
##
## Group means:
##   Auto.cylinders Auto.displacement Auto.horsepower Auto.weight
## 0       6.605839          266.1752       126.46715    3587.453
## 1       4.210145          116.5616        79.22464    2327.130
```

```
# Evaluate the classification performance of QDA.
my.thres <- 0.5
```

```
pred.qda.test <- predict(qda.mod, auto.test[,-1])
predicted.counterfeit.qda <- pred.qda.test$posterior[,'1'] > my.thres

# Confusion matrix.
table('Reference' = auto.test[,1] == '1', "Predicted" = predicted.counterfeit.qda)
```

```
##           Predicted
## Reference FALSE TRUE
##     FALSE    53    6
##     TRUE      4   54
```

```
test.error.qda <- 10/117

test.error.qda
```

```
## [1] 0.08547009
```

The test error of the QDA model is approximately 8.547%.

(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?

```
#Fit logistic regression model to training data.

m1 <- glm(mpg01~Auto.cylinders + Auto.displacement + Auto.horsepower + Auto.weight, data = auto.train,
          family = "binomial")
summary(m1)
```

```
##
## Call:
## glm(formula = mpg01 ~ Auto.cylinders + Auto.displacement + Auto.horsepower +
##     Auto.weight, family = "binomial", data = auto.train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.3830  -0.2362   0.0942   0.3273   2.8963
##
## Coefficients:
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)       12.5697852  2.0901467   6.014 1.81e-09 ***
## Auto.cylinders     0.3080626  0.4020183   0.766 0.443504
## Auto.displacement -0.0117588  0.0095732  -1.228 0.219334
## Auto.horsepower   -0.0425828  0.0171674  -2.480 0.013122 *
## Auto.weight       -0.0028528  0.0008453  -3.375 0.000738 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 381.23  on 274  degrees of freedom
## Residual deviance: 149.46  on 270  degrees of freedom
## AIC: 159.46
##
## Number of Fisher Scoring iterations: 7
```

```r
#Evaluate the classification performance of LR.
my.thres <- 0.5
pred.probs.test <- predict(m1, auto.test[,-1], type = 'response')
predicted.above <- pred.probs.test > my.thres
# Confusion matrix.
table('Reference' = auto.test[,1] == '1', "Predicted" = predicted.above)
```

```
##           Predicted
## Reference FALSE TRUE
##     FALSE    52    7
##     TRUE      5   53
```

```r
test.error.lr <- 12/117

test.error.lr
```

```
## [1] 0.1025641
```

The test error of the logistic regression model is approximately 10.256%.

(g) 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?

```r
library(class)

set.seed(123)

i <- 1:dim(auto)[1]

i.train <- sample(i,275, replace=F)

auto.train <- auto[i.train,-c(1,9)]

auto.test <- auto[-i.train, -c(1,9)]

above.train <- auto[i.train,]$mpg01
above.test <- auto[-i.train,]$mpg01

#Set K=1
m.knn <- knn(auto.train, auto.test, above.train, k = 1)
table(m.knn, above.test)
```

```
##       above.test
## m.knn  0  1
##     0 51 11
##     1  9 46
```

```r
test.error.1 <- 20/117
test.error.1
```

```
## [1] 0.1709402
```

```r
#Set K=2
m.knn2 <- knn(auto.train, auto.test, above.train, k = 2)
table(m.knn2, above.test)
```

```
##       above.test
```

```
## m.knn2  0  1
##      0 52  9
##      1  8 48
```

```
test.error.2 <- 17/117
test.error.2
```

```
## [1] 0.1452991
```

```
#Set K=10
m.knn3 <- knn(auto.train, auto.test, above.train, k = 10, prob = T)
table(m.knn3, above.test)
```

```
##       above.test
## m.knn3  0  1
##      0 54  4
##      1  6 53
```

```
test.error.10 <- 10/117
test.error.10
```

```
## [1] 0.08547009
```

For K=1, we get a test error rate of 17.094%. For K=2, we get a test error rate of 14.53%. For K=10, we get a test error rate of 8.547%. There is a clear decreasing trend in the test error rate as K increases, as a result for this data set it seems that a larger K value (>10) would preform best. However, K cannot be to large otherwise we will run into the curse of dimensionality as explored earlier.