Touray_Assignment3Resub

Sheikh-Sedat Touray

2023-09-23

We first call the the library function on **ISLR** to access the Auto dataset

```
library(ISLR)
Auto$origin = as.factor(Auto$origin)
summary(Auto)
```

```
##       mpg          cylinders      displacement     horsepower        weight
##  Min.   : 9.00   Min.   :3.000   Min.   : 68.0   Min.   : 46.0   Min.   :1613
##  1st Qu.:17.00   1st Qu.:4.000   1st Qu.:105.0   1st Qu.: 75.0   1st Qu.:2225
##  Median :22.75   Median :4.000   Median :151.0   Median : 93.5   Median :2804
##  Mean   :23.45   Mean   :5.472   Mean   :194.4   Mean   :104.5   Mean   :2978
##  3rd Qu.:29.00   3rd Qu.:8.000   3rd Qu.:275.8   3rd Qu.:126.0   3rd Qu.:3615
##  Max.   :46.60   Max.   :8.000   Max.   :455.0   Max.   :230.0   Max.   :5140
##
##   acceleration        year       origin                      name
##  Min.   : 8.00   Min.   :70.00   1:245   amc matador       :  5
##  1st Qu.:13.78   1st Qu.:73.00   2: 68   ford pinto        :  5
##  Median :15.50   Median :76.00   3: 79   toyota corolla    :  5
##  Mean   :15.54   Mean   :75.98           amc gremlin       :  4
##  3rd Qu.:17.02   3rd Qu.:79.00           amc hornet        :  4
##  Max.   :24.80   Max.   :82.00           chevrolet chevette:  4
##                                          (Other)           :365
```

a) Creating a new Variable **Origin2**

```
#create dummy variables for 1 if origin is American and 0 otherwise

origin2 <- ifelse (Auto$origin == "1", 1,0)
```

```
#change the datatype to numeric
origin2 <- as.numeric(origin2)
```

```
# Now we add the new column (origin2) to the Auto dataset
Auto2 <- cbind(Auto,origin2)
head(Auto2,5)
```

```
##   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
##                        name origin2
## 1 chevrolet chevelle malibu       1
## 2         buick skylark 320       1
```

```
## 3        plymouth satellite       1
## 4             amc rebel sst       1
## 5               ford torino       1
```

```r
# MASS library for the discriminant analysis models
library(MASS)
```

```r
library(caTools)

total_rows <- nrow(Auto2)

# Calculate the number of rows for the training and testing sets
train_rows <- round(0.8 * total_rows)  # 80% for training
test_rows <- total_rows - train_rows  # 20% for testing

# Generate random indices for the training set
train_indices <- sample(1:total_rows, train_rows)

# Create the training and testing datasets
train_data <- Auto2[train_indices, ]
test_data <- Auto2[-train_indices, ]
```

checking the dimensions of the train and test data.

```r
#viewing train data

dim(train_data)
```

```
## [1] 314  10
```

```r
#viewing test data

dim(test_data)
```

```
## [1] 78 10
```

c) Performing LDA on training data inorder to predict **origin2**

```r
# fitting the model on train data
modla2 <- lda(origin2 ~ acceleration + cylinders + horsepower + displacement, train_data)
modla2
```

```
## Call:
## lda(origin2 ~ acceleration + cylinders + horsepower + displacement,
##     data = train_data)
##
## Prior probabilities of groups:
##         0         1
## 0.3949045 0.6050955
##
## Group means:
##   acceleration cylinders horsepower displacement
## 0     16.51210  4.137097   79.82258     106.1774
## 1     14.97368  6.284211  119.01053     246.3474
##
## Coefficients of linear discriminants:
##                      LD1
## acceleration  0.006571392
```

```
## cylinders    -0.236066423
## horsepower   -0.023917522
## displacement  0.024639093
```

```r
# Predicting the lda model on the test data

lda.pred <- predict(modla2, test_data)$class
lda.pred
```

```
##  [1] 1 1 1 0 0 0 1 1 1 1 1 1 0 0 1 1 0 0 0 1 1 1 1 1 1 0 0 0 1 1 0 1 1 0 0 0 0 1
## [39] 1 1 0 0 1 1 1 1 1 1 1 1 1 0 0 1 1 0 0 1 1 1 1 1 0 0 1 1 0 0 0 1 0 0 0 1 0 0
## [77] 0 0
## Levels: 0 1
```

```r
#Creating a confusion matrix

table(lda.pred,test_data$origin2)
```

```
##
## lda.pred  0  1
##        0 22 12
##        1  1 43
```

```r
# checking the accuracy of the qda model
set.seed(19)
error_lda <- mean(lda.pred != test_data$origin2)
error_lda
```

```
## [1] 0.1666667
```

```r
# checking the accuracy of the lda model

acc_lda <- mean(lda.pred == test_data$origin2)
acc_lda
```

```
## [1] 0.8333333
```

d) Perform QDA on training data to predict **origin2**

```r
# fitting train data on model
modqa2 <- qda(origin2 ~ acceleration + cylinders + horsepower + displacement, train_data)
modqa2
```

```
## Call:
## qda(origin2 ~ acceleration + cylinders + horsepower + displacement,
##     data = train_data)
##
## Prior probabilities of groups:
##         0         1
## 0.3949045 0.6050955
##
## Group means:
##   acceleration cylinders horsepower displacement
## 0     16.51210  4.137097   79.82258     106.1774
## 1     14.97368  6.284211  119.01053     246.3474
```

```r
# Predicting the qda model on unseen data
qda.pred <- predict(modqa2, test_data)$class
qda.pred
```

```
##  [1] 1 1 1 0 0 1 1 1 1 1 1 1 0 0 1 1 0 0 0 1 1 1 1 1 1 1 0 0 1 1 0 1 1 0 0 0 0 1
## [39] 1 1 0 0 1 1 1 1 1 1 1 1 1 0 1 1 1 1 0 1 1 1 1 1 0 0 1 1 0 1 0 1 1 0 0 1 1 0
## [77] 0 0
## Levels: 0 1
```

```r
#creating a confusion matrix
table(qda.pred,test_data$origin2)
```

```
##
## qda.pred  0  1
##        0 19  8
##        1  4 47
```

```r
# checking for the error of the qda model
set.seed(17)
error_qda <- mean(qda.pred != test_data$origin2)
error_qda
```

```
## [1] 0.1538462
```

```r
# checking the accuracy of the qda model
acc_qda <- mean(qda.pred == test_data$origin2)
acc_qda
```

```
## [1] 0.8461538
```

The test error obtained from the QDA is less than that of the LDA is the same, I had different values before I used the set seed function but because random samples are generated if setseed function is not used these numbers are bound to change. In the first try QDA had a lower error but not by much

e) Perform KNN on the training data with several values of K to Predict **origin2**

Firstly, we import the class library

```r
library(class)
```

```r
set.seed(246)
#Prepare the data for training using the same variables in our previous models
train_knn <- cbind(train_data$cylinders, train_data$displacement, train_data$horsepower, train_data$acce

#Prepare the data for testing using the same variables in our previous models
test_knn<- cbind(test_data$cylinders, test_data$displacement, test_data$horsepower, test_data$accelerat

#use our test and train datasets to make a prediction for origin2
knn.pred <- knn(train_knn, test_knn, train_data$origin2, k = 1)

#create a confusion matrix for our prediction to easily show TP,TN,FP and FN
table(knn.pred, test_data$origin2)
```

```
##
## knn.pred  0  1
##        0 20  1
##        1  3 54
```

```r
# checking the accuracy of the k=1 model
```

```r
set.seed(12)
acc_knn <- mean(knn.pred == test_data$origin2)
acc_knn
```

Accuracy for K = 1

```
## [1] 0.9487179
```

```r
#use our test and train datasets to make a prediction for origin2
knn.pred <- knn(train_knn, test_knn, train_data$origin2, k = 3)

#create a confusion matrix for our prediction to easily show TP,TN,FP and FN
table(knn.pred, test_data$origin2)
```

For K = 3, we have...

```
##
## knn.pred  0  1
##        0 20  3
##        1  3 52
```

```r
# checking the accuracy of the k=3 model
set.seed(4)
acc_knn <- mean(knn.pred == test_data$origin2)
acc_knn
```

Accuracy for K = 3

```
## [1] 0.9230769
```

```r
#use our test and train datasets to make a prediction for origin2
knn.pred <- knn(train_knn, test_knn, train_data$origin2, k = 5)

#create a confusion matrix for our prediction to easily show TP,TN,FP and FN
table(knn.pred, test_data$origin2)
```

For K = 5, we have...

```
##
## knn.pred  0  1
##        0 20  4
##        1  3 51
```

```r
# checking the accuracy of the k=5 model
set.seed(5)
acc_knn <- mean(knn.pred == test_data$origin2)
acc_knn
```

Accuracy for K = 5

```
## [1] 0.9102564
```

```r
# checking the accuracy of the k=5 model
set.seed(5)
err_knn <- mean(knn.pred != test_data$origin2)
err_knn
```

```
## [1] 0.08974359
```

K=1 performs better than 3 and 5 with a higher accuracy before I used setseed function but after I used set seed function k=5 had the lowest error of about 0.038.

f) Use **5-fold CV** to Evaluate my best classifier.

```r
set.seed(1779)
lda.cv.error.5 <- rep(0,10) # essential to define a vector containing misclassification LDA errors for
knn.cv.error.5 <- rep(0,10) # essential to define a vector containing misclassification QDA errors for
n.test_data <- round(length(Auto$origin2)/10) # an approximate number of obrevations in each fold
n <- length(Auto2$origin2)
```

```r
for (i in 1:10){
test_data1 <- seq((i-1)*n.test_data+1,min(i*n.test_data,n))  # ordered test sequence
train_data1 <- setdiff(c(1:n),test_data) # ordered train sequence
set.seed(246)
#Prepare the data for training using the same variables in our previous models
train_knn <- cbind(Auto2$cylinders, Auto2$displacement, Auto2$horsepower, Auto2$acceleration)[train_dat

#Prepare the data for testing using the same variables in our previous models
test_knn<- cbind(Auto2$cylinders, Auto2$displacement, Auto2$horsepower, Auto2$acceleration)[test_data1,

train.y <- origin2[train_data1]

#use our test and train datasets to make a prediction for origin2
knn.pred1 <- knn(train_knn, test_knn, train.y, k = 5)

knn.cv.error.5[i]<- mean(knn.pred!=Auto2$origin2[test_data1])

#5 fold cv for lda
modla2 <- lda(origin2 ~ acceleration + cylinders + horsepower + displacement,train_data)
lda.pred <- predict(modla2, test_data)$class
lda.cv.error.5[i]<- mean(lda.pred != test_data$origin2)
}
```

The Knn errors after 10 tries and the average is.

```r
knn.cv.error.5
```

```
##  [1] 0.3076923 0.3076923 0.3076923 0.3076923 0.3076923 0.3076923 0.3076923
##  [8] 0.3076923 0.3076923 0.3076923
```

```r
mean(knn.cv.error.5)
```

```
## [1] 0.3076923
```

The LDA errors after 10 tries and the average is.

```r
lda.cv.error.5
```

```
##  [1] 0.1666667 0.1666667 0.1666667 0.1666667 0.1666667 0.1666667 0.1666667
```

```
##  [8] 0.1666667 0.1666667 0.1666667
```

```r
mean(lda.cv.error.5)
```

```
## [1] 0.1666667
```

g) Fit logistic regression with **origin2**

```r
auto2.fit<-glm(origin2~mpg +displacement+horsepower+weight+year+cylinders, data=train_data,family=binomi
summary(auto2.fit)
```

```
##
## Call:
## glm(formula = origin2 ~ mpg + displacement + horsepower + weight +
##     year + cylinders, family = binomial, data = train_data)
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -5.800268   5.272389  -1.100 0.271279
## mpg          -0.140833   0.067508  -2.086 0.036964 *
## displacement  0.121306   0.019383   6.258 3.89e-10 ***
## horsepower   -0.047440   0.018871  -2.514 0.011940 *
## weight       -0.004475   0.001170  -3.825 0.000131 ***
## year          0.199645   0.088610   2.253 0.024254 *
## cylinders    -1.434479   0.511379  -2.805 0.005030 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 421.32  on 313  degrees of freedom
## Residual deviance: 159.13  on 307  degrees of freedom
## AIC: 173.13
##
## Number of Fisher Scoring iterations: 8
```

All the predictors seem to be statistically significant.

mpg, weight and cylinders have a negative coefficient meaning if they go down then the origin of the car is not American and if they go up then the car is American.

While year and displacement have positive coefficients and the opposite of the statement above is the truth for them.

(h) Obtain a prediction of origin2 status for each car by computing the posterior probability of being manufactured in American.

```r
auto2.probs = predict(auto2.fit, test_data, type = "response")
auto2.pred = rep(0, length(auto2.probs))
auto2.pred[auto2.probs > 0.5] = 1
```

(i) Compute the validation set error, which is the fraction of the observations in the 20% validation set that are misclassified.

```r
table(auto2.pred, test_data$origin2)
```

```
##
## auto2.pred  0  1
```

```
##          0 22  8
##          1  1 47
```

```
# checking the accuracy of the k=5 model
err_aut2 <- mean(auto2.pred != test_data$origin2)
err_aut2
```

```
## [1] 0.1153846
```

The error in *glm* is much lower than all the other previous classifiers and it seems to be much better that all the other classifiers.

J) Fit logistic regression with **origin2** as the response and mpg and horsepower as predictors on the full dataset.

```
set.seed(3)
auto21.fit<-glm(origin2 ~ mpg + horsepower, data=Auto2,family=binomial)
summary(auto21.fit)
```

```
##
## Call:
## glm(formula = origin2 ~ mpg + horsepower, family = binomial,
##     data = Auto2)
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  1.254207   1.283700   0.977   0.3286
## mpg         -0.125032   0.028043  -4.459 8.25e-06 ***
## horsepower   0.024441   0.007697   3.176   0.0015 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 518.67  on 391  degrees of freedom
## Residual deviance: 366.71  on 389  degrees of freedom
## AIC: 372.71
##
## Number of Fisher Scoring iterations: 5
```

K) Write a function, **boot.fn()** that takes

```
library(boot)
library(lattice)
```

```
##
## Attaching package: 'lattice'

## The following object is masked from 'package:boot':
##
##     melanoma
```

```
boot.fn <- function(data, index)
  return(coef(lm(origin2 ~ mpg + horsepower, data = data, subset = index)))
boot.fn(Auto2, 1:392)
```

```
## (Intercept)         mpg  horsepower
##  1.13912577 -0.02900395  0.00158801
```

l) use the **boot()** function with the **boot.fn()** function to estimate

```
boot.fn <- function(data, index)
  coefficients(lm(origin2 ~ mpg + horsepower, data = data, subset = index))

set.seed(1)
boot(Auto2, boot.fn, 1000)
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = Auto2, statistic = boot.fn, R = 1000)
##
##
## Bootstrap Statistics :
##        original        bias     std. error
## t1*  1.13912577 -9.133301e-03 0.1553744607
## t2* -0.02900395  2.209689e-04 0.0038111238
## t3*  0.00158801  3.438073e-05 0.0006549195
```

The standard errors obtained from the bootstrap function are much lower than those obtained from the **glm()** function.