**Assignment #4: Classification & Cross Validation**

**Submit through link: eCampus -> Assignments->Homework 4 Submission**

**Deadline: October 25 (Friday) 5:00 pm**

**The filename should have this format: LastName-FirstName-hw04.doc**

**Problem 1 (14pt)**

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

(a) Create a binary variable, mpg01, that contains a 1 if mpgcontains a value above its median, and a 0 if mpgcontains a value below its median. You can compute the median using the median( )function. Note that you may find it helpful to use the data.frame( )function to create a single data set containing both mpg01and the other Autovariables.

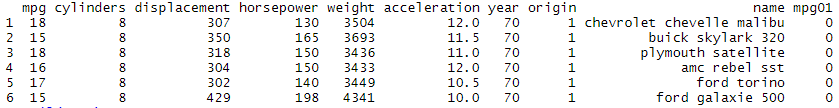
> library(ISLR)

> mpg01 <- rep(0, length(Auto$mpg))

> mpg01[Auto$mpg > median(Auto$mpg)] <- 1

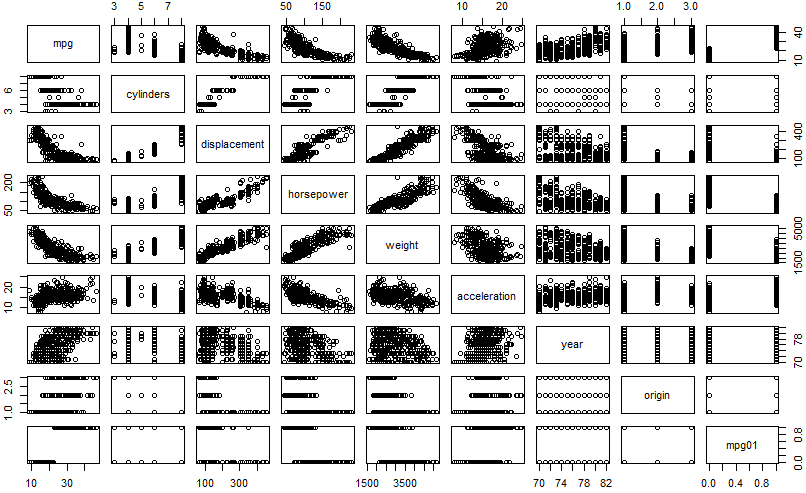
> Auto <- data.frame(Auto, mpg01)

> head(Auto)



(b) Explore the data graphically in order to investigate the association between mpg01and 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[, -9])



> par(mfrow=c(2,3))

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

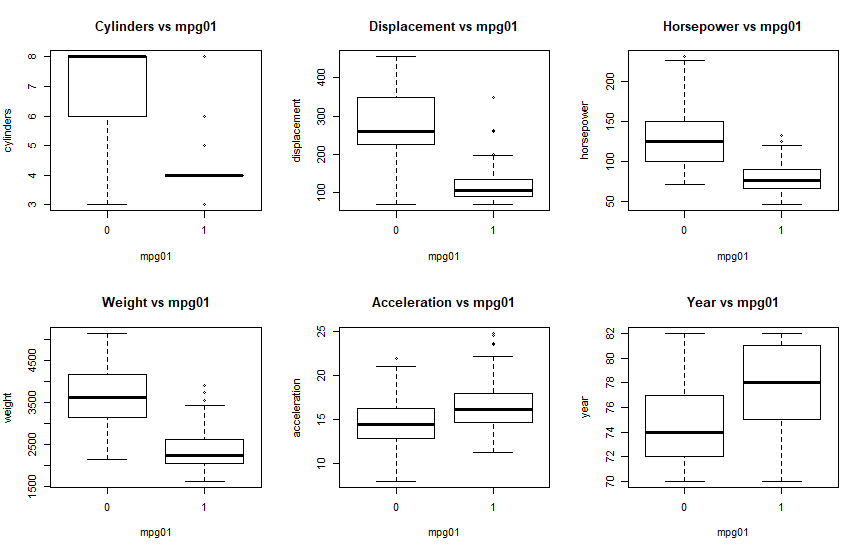
> boxplot(displacement ~ mpg01, data = Auto, main = "Displacement vs mpg01")

> 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")



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

(d) Perform LDA on the training data in order to predict mpg01using the variables that seemed most associated with mpg01 in (b). What is the test error of the model obtained?

**Creating the model:**

> lda\_model <- lda(mpg01 ~ cylinders + weight + displacement + horsepower, data = training\_data)

> lda\_model

Call:

lda(mpg01 ~ cylinders + weight + displacement + horsepower, data = training\_data)

Prior probabilities of groups:

0 1

0.4963504 0.5036496

Group means:

cylinders weight displacement horsepower

0 6.786765 3641.022 275.2941 130.96324

1 4.188406 2314.000 114.5290 78.00725

Coefficients of linear discriminants:

LD1

cylinders -0.3974647924

weight -0.0009670704

displacement -0.0029615583

horsepower 0.0049004106

**Computing the confusion Matrix:**

> lda\_pred = predict(lda\_model, testing\_data)

> pred.lda <- predict(lda\_model, testing\_data)

> table(pred.lda$class, mpg01.test)

0 1

0 50 3

1 10 55

**Calculating the test error:**

> mean(pred.lda$class != mpg01.test)

[1] 0.1101695

Thus we conclude that we have a mean error rate of 11.02% with LDA.

(e) Perform QDA on the training data in order to predict mpg01using the variables that seemed most associated with mpg01in (b). What is the test error of the model obtained?

**Creating the model:**

> qda\_model = qda(mpg01 ~ cylinders + horsepower + weight + acceleration, data=training\_data)

> qda\_model

Call:

qda(mpg01 ~ cylinders + horsepower + weight + acceleration, data = training\_data)

Prior probabilities of groups:

0 1

0.4963504 0.5036496

Group means:

cylinders horsepower weight acceleration

0 6.786765 130.96324 3641.022 14.55588

1 4.188406 78.00725 2314.000 16.55072

**Computing the confusion matrix:**

> qda.class=predict(qda\_model, testing\_data)$class

> table(qda.class, testing\_data$mpg01)

qda.class 0 1

0 53 4

1 7 54

**Calculating the Test Error:**

> mean(qda.class != testing\_data$mpg01)

[1] 0.09322034

We conclude that the test error of QDA is 9.32%.

(f) Perform logistic regression on the training data in order to predict mpg01using the variables that seemed most associated with mpg01in (b). What is the test error of the model obtained?

**Creating the model:**

> glm\_model <- glm(mpg01 ~ cylinders + weight + displacement + horsepower, data = training\_data, family = binomial)

> summary(glm\_model)

Call:

glm(formula = mpg01 ~ cylinders + weight + displacement + horsepower,

family = binomial, data = training\_data)

Deviance Residuals:

Min 1Q Median 3Q Max

-2.44120 -0.17870 0.08712 0.31147 3.05303

Coefficients:

Estimate Std. Error z value Pr(>|z|)

(Intercept) 11.8103006 2.0819718 5.673 1.41e-08 \*\*\*

cylinders 0.1869071 0.3972245 0.471 0.63797

weight -0.0020251 0.0008573 -2.362 0.01817 \*

displacement -0.0164493 0.0095899 -1.715 0.08629 .

horsepower -0.0443408 0.0172072 -2.577 0.00997 \*\*

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 379.83 on 273 degrees of freedom

Residual deviance: 138.27 on 269 degrees of freedom

AIC: 148.27

Number of Fisher Scoring iterations: 7

**Computing the confusion matrix:**

> probs <- predict(glm\_model, testing\_data, type = "response")

> pred.glm <- rep(0, length(probs))

> pred.glm[probs > 0.5] <- 1

> table(pred.glm, mpg01.test)

mpg01.test

pred.glm 0 1

0 53 6

1 7 52

**Calculating the Test Error:**

> mean(pred.glm != mpg01.test)

[1] 0.1101695

We conclude that the test error with Logistic Regression is 11.02% same as LDA

(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 mpg01in (b). What test errors do you obtain? Which value of *K* seems to perform the best on this data set?

**Splitting data for KNN:**

> data = scale(Auto[,-c(9,10)])

> set.seed(1234)

> train <- sample(1:dim(Auto)[1], 392\*.7, rep=FALSE)

> #train <- sample(1:dim(Auto)[1], dim(Auto)[1]\*.7, rep=FALSE)

> test <- -train

> training\_data = data[train,c("cylinders","horsepower","weight","acceleration")]

> testing\_data = data[test, c("cylinders", "horsepower","weight","acceleration")]

> ## KNN take the training response variable seperately

> train.mpg01 = Auto$mpg01[train]

>

> test.mpg01= Auto$mpg01[test]

**Computing the confusion matrix:**

> library(class)

> set.seed(1234)

> knn\_pred\_y = knn(training\_data, testing\_data, train.mpg01, k = 1)

> table(knn\_pred\_y, test.mpg01)

test.mpg01

knn\_pred\_y 0 1

0 57 5

1 7 49

**Calculating the Test Error (k=1):**

> mean(knn\_pred\_y != test.mpg01)

[1] 0.1016949

**Calculating the optimum value of k and its corresponding test error:**

> knn\_pred\_y = NULL

> error\_rate = NULL

> for(i in 1:dim(testing\_data)[1]){

+ set.seed(1234)

+ knn\_pred\_y = knn(training\_data,testing\_data,train.mpg01,k=i)

+ error\_rate[i] = mean(test.mpg01 != knn\_pred\_y)

+ }

>

> ### find the minimum error rate

> min\_error\_rate = min(error\_rate)

> print(min\_error\_rate)

[1] 0.09322034

> ### get the index of that error rate, which is the k

> K = which(error\_rate == min\_error\_rate)

> print(K)

[1] 4

We conclude that the optimum value of k was k=4 which gave us the lowest test error of 9.3%

**Problem 2 (6pt)**

Perform ROC analysis and present the results for LDA and KNN for a given K. Use the same model in Problem 1.

**LDA**

> lda.fit.post <- as.data.frame(lda.predict$posterior)

> pred <- prediction(lda.fit.post[,2], Auto.test$mpg01)

> roc\_perf = performance(pred, measure = "tpr", x.measure = "fpr")

> auc.train <- performance(pred,measure = "auc")

> auc.train <- auc.train@y.values

> plot(roc\_perf)

> abline(a=0,b=1)

> text(x =0.25, y =0.65, paste("AUC =",round(auc.train[[1]],3), sep = ""))

A close up of a map

Description automatically generated

KNN = 1

predknn = prediction(as.double(knn.pred),Auto.test$mpg01)

> roc1\_perf = performance(predknn, measure = "tpr", x.measure = "fpr")

> auc.train1 <- performance(predknn,measure = "auc")

> auc.train1 <- auc.train1@y.values

> plot(roc1\_perf)

> abline(a=0, b = 1)

> text(x =0.25 , y = 0.65, paste("AUC = ", round(auc.train[[1]],3), sep = ""))

A close up of a map

Description automatically generated

KNN = 3

> predknn3 = prediction(as.double(knn.pred3),Auto.test$mpg01)

> roc1\_perf3 = performance(predknn3, measure = "tpr", x.measure = "fpr")

> auc.train3 <- performance(predknn3,measure = "auc")

> auc.train3 <- auc.train3@y.values

> plot(roc1\_perf3)

> abline(a=0, b = 1)

> text(x =0.25 , y = 0.65, paste("AUC = ", round(auc.train[[1]],3), sep = ""))

A close up of a map

Description automatically generated

K =5

> predknn5 = prediction(as.double(knn.pred5),Auto.test$mpg01)

> roc1\_perf5 = performance(predknn5, measure = "tpr", x.measure = "fpr")

> auc.train5 <- performance(predknn5,measure = "auc")

> auc.train5 <- auc.train5@y.values

> plot(roc1\_perf5)

> abline(a=0, b = 1)

> text(x =0.25 , y = 0.65, paste("AUC = ", round(auc.train[[1]],3), sep = ""))

A close up of a map

Description automatically generated

**Problem 3 (8pt)**

This question should be answered using the Default data set. In Chapter 4 on classification, we used logistic regression to predict the probability of default using income and balance. Now we will estimate the test error of this logistic regression model using the validation set approach. Do not forget to set a random seed before beginning your analysis.

(a) Fit a logistic regression model that predicts default using income and balance.

> library(ISLR)

> attach(Default)

> set.seed(1)

> fit.glm <- glm(default ~ income + balance, data = Default, family = "binomial")

> summary(fit.glm)

Call:

glm(formula = default ~ income + balance, family = "binomial",

data = Default)

Deviance Residuals:

Min 1Q Median 3Q Max

-2.4725 -0.1444 -0.0574 -0.0211 3.7245

Coefficients:

Estimate Std. Error z value Pr(>|z|)

(Intercept) -1.154e+01 4.348e-01 -26.545 < 2e-16 \*\*\*

income 2.081e-05 4.985e-06 4.174 2.99e-05 \*\*\*

balance 5.647e-03 2.274e-04 24.836 < 2e-16 \*\*\*

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 2920.6 on 9999 degrees of freedom

Residual deviance: 1579.0 on 9997 degrees of freedom

AIC: 1585

Number of Fisher Scoring iterations: 8

(b) Using the validation set approach, estimate the test error of this model. You need to perform the following steps:

i. Split the sample set into a training set and a validation set.

> train <- sample(dim(Default)[1], dim(Default)[1] / 2)

ii. Fit a logistic regression model using only the training data set.

> fit.glm <- glm(default ~ income + balance, data = Default, family = "binomial", subset = train)

> summary(fit.glm)

Call:

glm(formula = default ~ income + balance, family = "binomial",

data = Default, subset = train)

Deviance Residuals:

Min 1Q Median 3Q Max

-2.5830 -0.1428 -0.0573 -0.0213 3.3395

Coefficients:

Estimate Std. Error z value Pr(>|z|)

(Intercept) -1.194e+01 6.178e-01 -19.333 < 2e-16 \*\*\*

income 3.262e-05 7.024e-06 4.644 3.41e-06 \*\*\*

balance 5.689e-03 3.158e-04 18.014 < 2e-16 \*\*\*

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 1523.8 on 4999 degrees of freedom

Residual deviance: 803.3 on 4997 degrees of freedom

AIC: 809.3

Number of Fisher Scoring iterations: 8

iii. Obtain a prediction of default status for each individual in the validation set using a threshold of 0.5.

> probs <- predict(fit.glm, newdata = Default[-train, ], type = "response")

> pred.glm <- rep("No", length(probs))

> pred.glm[probs > 0.5] <- "Yes"

iv. Compute the validation set error, which is the fraction of the observations in the validation set that are misclassified.

> mean(pred.glm != Default[-train, ]$default)

[1] 0.0254

We have a test error rate of 2.54%

(c) Repeat the process in (b) three times, using three different splits of the observations into a training set and a validation set. Comment on the results obtained.

> train <- sample(dim(Default)[1], dim(Default)[1] / 2)

> fit.glm <- glm(default ~ income + balance, data = Default, family = "binomial", subset = train)

> probs <- predict(fit.glm, newdata = Default[-train, ], type = "response")

> pred.glm <- rep("No", length(probs))

> pred.glm[probs > 0.5] <- "Yes"

> mean(pred.glm != Default[-train, ]$default)

[1] 0.0274

> train <- sample(dim(Default)[1], dim(Default)[1] / 2)

> fit.glm <- glm(default ~ income + balance, data = Default, family = "binomial", subset = train)

> probs <- predict(fit.glm, newdata = Default[-train, ], type = "response")

> pred.glm <- rep("No", length(probs))

> pred.glm[probs > 0.5] <- "Yes"

> mean(pred.glm != Default[-train, ]$default)

[1] 0.0244

> train <- sample(dim(Default)[1], dim(Default)[1] / 2)

> fit.glm <- glm(default ~ income + balance, data = Default, family = "binomial", subset = train)

> probs <- predict(fit.glm, newdata = Default[-train, ], type = "response")

> pred.glm <- rep("No", length(probs))

> pred.glm[probs > 0.5] <- "Yes"

> mean(pred.glm != Default[-train, ]$default)

[1] 0.0244

Since we randomly sample the data in the training and validation set, the test error can change slightly based on the sample but will more or less be close to each other.

(d) Consider another logistic regression model that predicts default using income, balance and student (qualitative). Estimate the test error for this model using the validation set approach. Does including the qualitative variable student lead to a reduction of test error rate?

> train <- sample(dim(Default)[1], dim(Default)[1] / 2)

> fit.glm <- glm(default ~ income + balance + student, data = Default, family = "binomial", subset = train)

> pred.glm <- rep("No", length(probs))

> probs <- predict(fit.glm, newdata = Default[-train, ], type = "response")

> pred.glm[probs > 0.5] <- "Yes"

> mean(pred.glm != Default[-train, ]$default)

[1] 0.0264

From above, we observe that the test error rate does not change much with the addition of the dummy variable.

**Problem 4 (14pt)**

This question requires performing cross validation on a simulated data set.

(a) Generate a simulated data set as follows:

set.seed(1)

x=rnorm(200)

y=x-2\*x^2+rnorm(200)

In this data set, what is and what is ? Write out the model used to generate the data in equation form (i.e., the true model of the data).

> set.seed(1)

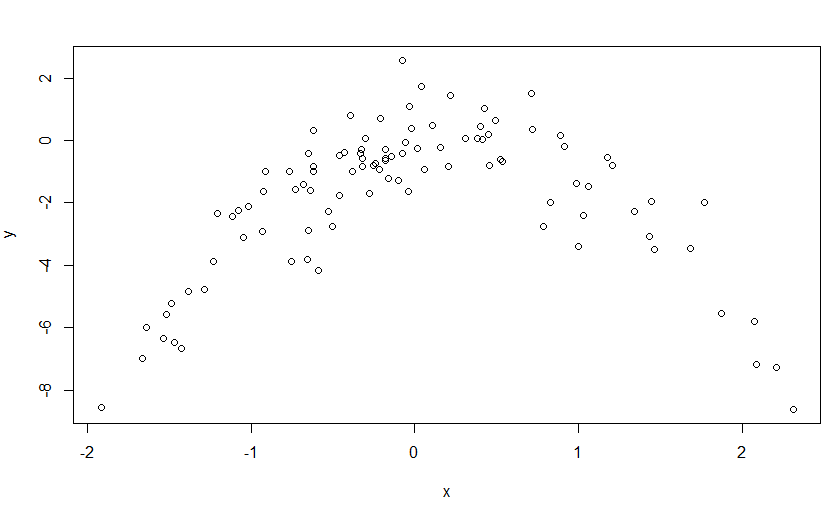
> y <- rnorm(100)

> x <- rnorm(100)

> y <- x - 2 \* x^2 + rnorm(100)

(b) Create a scatter plot of vs . Comment on what you find.

> plot(x, y)



(c) Consider the following four models for the data set:

i.

> library(boot)

> set.seed(1)

> Data <- data.frame(x, y)

> fit.glm.1 <- glm(y ~ x)

> cv.glm(Data, fit.glm.1)$delta[1]

[1] 5.890979

ii.

> fit.glm.2 <- glm(y ~ poly(x, 2))

> cv.glm(Data, fit.glm.2)$delta[1]

[1] 1.086596

iii.

> fit.glm.3 <- glm(y ~ poly(x, 3))

> cv.glm(Data, fit.glm.3)$delta[1]

[1] 1.102585

iv.

> fit.glm.4 <- glm(y ~ poly(x, 4))

> cv.glm(Data, fit.glm.4)$delta[1]

[1] 1.114772

Compute the LOOCV errors that result from fitting these models.

(d) Repeat (c) using another random seed, and report your results. Are your results the same as what you got in (c)? Why?

> set.seed(10)

> fit.glm.1 <- glm(y ~ x)

> cv.glm(Data, fit.glm.1)$delta[1]

[1] 5.890979

> fit.glm.2 <- glm(y ~ poly(x, 2))

> cv.glm(Data, fit.glm.2)$delta[1]

[1] 1.086596

> fit.glm.3 <- glm(y ~ poly(x, 3))

> cv.glm(Data, fit.glm.3)$delta[1]

[1] 1.102585

> fit.glm.4 <- glm(y ~ poly(x, 4))

> cv.glm(Data, fit.glm.4)$delta[1]

[1] 1.114772

(e) Which of the models in (c) had the smallest LOOCV error? Is this what you expected? Explain your answer.

We may see that the LOOCV estimate for the test MSE is minimum for “fit.glm.2”, this is not surprising since we saw clearly in (b) that the relation between “x” and “y” is quadratic.

(f) Now we use 5-fold CV for the model selection. Compute the CV errors that result from fitting the four models. Which model has the smallest CV error? Are the results consistent with LOOCV?

> cv.glm(Data, fit.glm.1, K=5)$delta[1]

[1] 6.617861

> cv.glm(Data, fit.glm.2, K=5)$delta[1]

[1] 1.076019

> cv.glm(Data, fit.glm.3, K=5)$delta[1]

[1] 1.110618

> cv.glm(Data, fit.glm.4, K=5)$delta[1]

[1] 1.135411

The Quadratic model has the smallest 5 fold CV of all the four models. This is in consistent with the results of LOOCV because the CV errors were increasing on increase the degree of the polynomial. The reason for such low error is because the quadratic model accurately fits our data as checked in question b.

(g) Repeat (f) using 10-fold CV. Are the results the same as 5-fold CV?

> cv.glm(Data, fit.glm.1, K=10)$delta[1]

[1] 5.987802

> cv.glm(Data, fit.glm.2, K=10)$delta[1]

[1] 1.087681

> cv.glm(Data, fit.glm.3, K=10)$delta[1]

[1] 1.127126

> cv.glm(Data, fit.glm.4, K=10)$delta[1]

[1] 1.148407

The Quadratic model has the smallest 1o fold CV of all the four models and it is consistent with 5 fold. This consistency in the results of LOOCV is because the CV errors were increasing on increase the degree of the polynomial. The reason for such low error is because the quadratic model accurately fits our data as checked in question b.