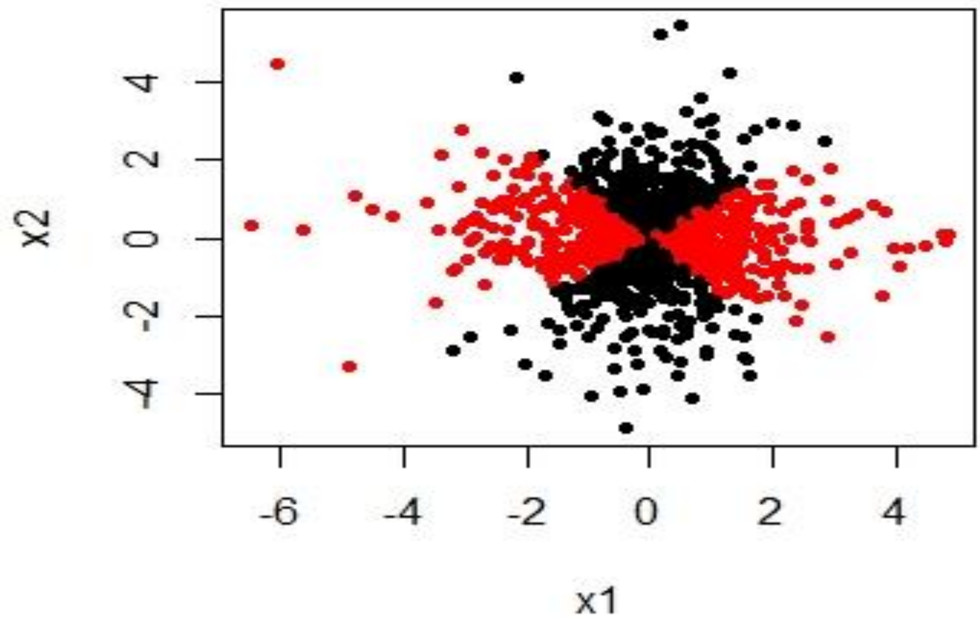
ECON7333 Assignment 2

SHENG-YI HUANG 45193837

Classification exercise

1. Plot the data on a figure

```
> # Assignment 2 SHENG-YI HUANG 45193837
>
> setwd("C:/Users/sei19/Desktop/7333/AS2")
> As2 =read.csv("As2.csv")
>
> library(MASS) #has the lda( function)
> library(class)
> library(leaps)
> library(boot) # use for cross validation
> attach(As2)
>
>
> # Q1 Plot the dot on a figure
> plot(x1,x2,col=factor(y), pch=20)
```



2. Fit a linear model to the data

```
> # Q2 Fit a linear model to a data
> # linear regression
> lm.fits=lm(y ~ x1 + x2)
>
> # predict the prob of y
> lm.probs = predict(lm.fits, type = "response")
>
> # predict for each of the x in our data set (fitted values)
> lm.pred = rep("0", 1000)
> lm.pred[lm.probs> .5]= "1"
> table(lm.pred, y) # compare our prediction to actural data
       y
lm.pred   0   1
      0  11  21
      1 462 506
> sum(y==lm.pred)/1000
[1] 0.517
```

Linear regression correctly predicted the y at 51.7%,

1 - 51.7% = 48.3% is the training error rate.

3. Repeat the same exercise for each of the following:

a. Logistic regression

```
> # Q3 Fit 3 models to a data
> # 3a Logistic regression
> glm.fits = glm(y ~ x1 + x2,
+                 data = As2, family = binomial)
>
> # predict the prob of y
> glm.probs = predict(glm.fits, type = "response")
>
> # predict for each of the x in our data set (fitted values)
> glm.pred = rep("0", 1000)
> glm.pred[glm.probs> .5]= "1"
> table(glm.pred, y)# lets compare our prediction to actural data
        y
glm.pred   0   1
       0  11  21
       1 462 506
> sum(y==glm.pred)/1000
[1] 0.517
```

Logistic regression correctly predicted the y at 51.7%,

1 - 51.7% = 48.3% is the training error rate.

b. Linear Discriminant Analysis

```
> # 3b Linear Discrminant Analysis
> lda.fits = lda(y ~ x1 + x2, data= As2)
>
> # predict with lda
> lda.pred = predict(lda.fits, As2)
> lda.class = lda.pred$class
> table(lda.class, y)
         y
lda.class   0   1
        0  11  21
        1 462 506
> mean(lda.class == y)
[1] 0.517
```

Linear discriminant analysis correctly predicted the y 51.7%,

1 - 51.7% = 48.3% is the training error rate.

c. Quadratic Discriminant Analysis

```
> # 3c Quadratic Discriminant Analysis
> qda.fits = qda(y ~ x1 + x2, data = As2)
>
> # predict with qda
> qda.pred = predict(qda.fits, As2)
> qda.class = qda.pred$class
> table(qda.class, y)
         y
qda.class   0   1
        0 418   0
        1  55 527
>
> mean(qda.class == y)
[1] 0.945
```

Quadratic discriminant analysis correctly predicted the y 94.5%,

1 - 94.5%= 5.5% is the training error rate.

4. Fit the model by k-nearest neighbor classification with k=1 – 20

```
> # Q4 Fit the model by knn
> # knn
> As2.X = scale(As2[,-3])
>
> # test = train
> set.seed(1)
> knn.pred = knn(As2.X , As2.X ,y, k=1)
> table(knn.pred, y)
        y
knn.pred   0   1
       0 473   0
       1   0 527
> mean(knn.pred == y)
[1] 1
>
> knn.pred = knn(As2.X , As2.X ,y, k=2)
> table(knn.pred, y)
        y
knn.pred   0   1
       0 468   9
       1   5 518
> mean(knn.pred == y)
[1] 0.986
>
> knn.pred = knn(As2.X , As2.X ,y, k=3)
> table(knn.pred, y)
        y
knn.pred   0   1
       0 466   3
       1   7 524
> mean(knn.pred == y)
[1] 0.99
>
> knn.pred = knn(As2.X , As2.X ,y, k=4)
> table(knn.pred, y)
        y
knn.pred   0   1
       0 461   5
       1  12 522
> mean(knn.pred == y)
[1] 0.983
>
> knn.pred = knn(As2.X , As2.X ,y, k=5)
> table(knn.pred, y)
        y
knn.pred   0   1
       0 465   9
       1   8 518
> mean(knn.pred == y)
[1] 0.983
>
> knn.pred = knn(As2.X , As2.X ,y, k=6)
> table(knn.pred, y)
        y
knn.pred   0   1
       0 463  10
       1  10 517
> mean(knn.pred == y)
[1] 0.98
>
> knn.pred = knn(As2.X , As2.X ,y, k=7)
> table(knn.pred, y)
        y
knn.pred   0   1
       0 465   8
       1   8 519
```

```
> mean(knn.pred == y)
[1] 0.984
>
> knn.pred = knn(As2.X , As2.X ,y, k=8)
> table(knn.pred, y)
        y
knn.pred   0   1
       0 466  11
       1   7 516
> mean(knn.pred == y)
[1] 0.982
>
> knn.pred = knn(As2.X , As2.X ,y, k=9)
> table(knn.pred, y)
        y
knn.pred   0   1
       0 465  11
       1   8 516
> mean(knn.pred == y)
[1] 0.981
>
> knn.pred = knn(As2.X , As2.X ,y, k=10)
> table(knn.pred, y)
        y
knn.pred   0   1
       0 466  11
       1   7 516
> mean(knn.pred == y)
[1] 0.982
>
> knn.pred = knn(As2.X , As2.X ,y, k=11)
> table(knn.pred, y)
        y
knn.pred   0   1
       0 467  11
       1   6 516
> mean(knn.pred == y)
[1] 0.983
>
> knn.pred = knn(As2.X , As2.X ,y, k=12)
> table(knn.pred, y)
        y
knn.pred   0   1
       0 466  12
       1   7 515
> mean(knn.pred == y)
[1] 0.981
>
> knn.pred = knn(As2.X , As2.X ,y, k=13)
> table(knn.pred, y)
        y
knn.pred   0   1
       0 466   8
       1   7 519
> mean(knn.pred == y)
[1] 0.985
>
> knn.pred = knn(As2.X , As2.X ,y, k=14)
> table(knn.pred, y)
        y
knn.pred   0   1
       0 462   8
       1  11 519
> mean(knn.pred == y)
[1] 0.981
>
```

```
> knn.pred = knn(As2.X , As2.X ,y, k=15)
> table(knn.pred, y)
        y
knn.pred   0   1
       0 462   3
       1  11 524
> mean(knn.pred == y)
[1] 0.986
>
> knn.pred = knn(As2.X , As2.X ,y, k=16)
> table(knn.pred, y)
        y
knn.pred   0   1
       0 462   3
       1  11 524
> mean(knn.pred == y)
[1] 0.986
>
> knn.pred = knn(As2.X , As2.X ,y, k=17)
> table(knn.pred, y)
        y
knn.pred   0   1
       0 460   6
       1  13 521
> mean(knn.pred == y)
[1] 0.981
>
> knn.pred = knn(As2.X , As2.X ,y, k=18)
> table(knn.pred, y)
        y
knn.pred   0   1
       0 460   7
       1  13 520
> mean(knn.pred == y)
[1] 0.98
>
> knn.pred = knn(As2.X , As2.X ,y, k=19)
> table(knn.pred, y)
        y
knn.pred   0   1
       0 461   6
       1  12 521
> mean(knn.pred == y)
[1] 0.982
>
> knn.pred = knn(As2.X , As2.X ,y, k=20)
> table(knn.pred, y)
        y
knn.pred   0   1
       0 460   8
       1  13 519
> mean(knn.pred == y)
[1] 0.979
```

Table for KNN from 1 to 20, the accurate rate and error rate:

K	1	2	**3**	4	5	6	7	8	9	10
Accurate	1	0.986	**0.99**	0.983	0.983	0.98	0.984	0.982	0.981	0.982
Error	0	0.014	**0.01**	0.017	0.017	0.02	0.016	0.018	0.019	0.018
K	11	12	13	14	**15**	**16**	17	18	19	20
Accurate	0.983	0.981	**0.985**	0.981	**0.986**	**0.986**	0.981	0.98	0.982	0.979
Error	0.017	0.019	**0.015**	0.019	**0.014**	**0.014**	0.019	0.02	0.018	0.021

5. Choose between all 24 methods by using 10-fold cross-validation. Try to justify the results based on your intuition regarding the data.

a. Linear regression 10-fold cross-validation

```
> # Q5 24 methods by using 10-fold cross-validation
> # k fold cross validation
> # 5a lm 10 fold ok
> k=10
> set.seed(1)
> # divide data into 10 groups
> folds=sample(1:k, nrow(As2), replace=T) # divide data into 10 grou
>
> cv.errors.lm=matrix(0,k,2)#matrix of zeros for storage of MSEs
> # we want to loop over the 2 models also over 10 folds
> #train on everything except j fold
> for(j in 1:k){
+   regfit.best=regsubsets(y ~ ., data = As2[folds!=j,],nvmax=2)
+   test.mat=model.matrix(y ~.,data=As2[folds==j,])
+   #we used to predit test data in jth fold
+   for(i in 1:2){
+     coefi= coef(regfit.best, id=i)
+     pred=test.mat[,names(coefi)]%*%coefi
+     cv.errors.lm[j,i]=mean((pred-As2$y[folds==j])^2)#estimate MSE
+   }
+ }
> mean.cv.errors.lm = mean(apply(cv.errors.lm,1,mean))
> mean.cv.errors.lm
[1] 0.2507484
```

Estimated MSE is 0.2507.

b. Logistic regression 10-fold cross-validation

```
> # 5b glm 10 fold ok
> cv.errors.glm = rep(0,10)
> for(i in 1:10){
+ cv.errors.glm[i] = cv.glm(As2, glm.fits, K=10)$delta[1]
+ }
> mean(cv.errors.glm)
[1] 0.2505784
```

Estimated MSE is 0.2505.

c. Linear discrimination analysis 10-fold cross-validation

```
> # 5c lda 10 fold
> K=10
> # use sapply() to calculate 10 fold MSE
> cv.lda = sapply(1:K, FUN=function(i){
+   testID = which(folds == i, arr.ind = TRUE)
+   test = As2[testID,]
+   train = As2[-testID,]
+   ldaf = lda(y ~ ., data = train)
+   lda.pred.test = predict(ldaf, test)
+   cv.est.lda = mean(lda.pred.test$class != test$y) # MSE
+   return(cv.est.lda)
+ })
> mean(cv.lda)
[1] 0.4916111
```

Estimated MSE is 0.4916.

d. Quadradic discrimination analysis 10-fold cross-validation

```
> # 5d qla 10 fold cross-validation
> K=10
> cv.qda = sapply(1:K, FUN=function(i){
+   testID = which(folds == i, arr.ind = TRUE)
+   test = As2[testID,]
+   train = As2[-testID,]
+   qdaf = qda(y ~ ., data = train)
+   qda.pred.test = predict(qdaf, test)
+   cv.est.qda = mean(qda.pred.test$class != test$y) # MSE
+   return(cv.est.qda)
+ })
> mean(cv.qda)
[1] 0.06517976
```

Estimated MSE is 0.6517.

e. KNN 10-fold cross-validation

```
> # 5e knn 10 fold cross-validation
> # knn = 1
> K=10
> set.seed(1)
> cv.knn = sapply(1:K, FUN=function(i){
+   testID = which(folds == i, arr.ind = TRUE)
+   test = As2[testID,]
+   train = As2[-testID,]
+
+   knn.pred.test = knn(train[,1:2], test[,1:2], cl=train$y,k=1)
+   cv.est.knn = mean(knn.pred.test != test$y)
+   return(cv.est.knn)
+ })
> mean(cv.knn)
[1] 0.02748032
>
```

```
> # knn = 3
> K=10
> cv.knn = sapply(1:K, FUN=function(i){
+   testID = which(folds == i, arr.ind = TRUE)
+   test = As2[testID,]
+   train = As2[-testID,]
+
+   knn.pred.test = knn(train[,1:2], test[,1:2], cl=train$y,k=3)
+   cv.est.knn = mean(knn.pred.test != test$y)
+   return(cv.est.knn)
+ })
> mean(cv.knn)
[1] 0.03059054
>
> K=10
> cv.knn = sapply(1:K, FUN=function(i){
+   testID = which(folds == i, arr.ind = TRUE)
+   test = As2[testID,]
+   train = As2[-testID,]
+
+   knn.pred.test = knn(train[,1:2], test[,1:2], cl=train$y,k=13)
+   cv.est.knn = mean(knn.pred.test != test$y)
+   return(cv.est.knn)
+ })
> mean(cv.knn)
[1] 0.01949453
>
> # knn = 15
> K=10
> cv.knn = sapply(1:K, FUN=function(i){
+   testID = which(folds == i, arr.ind = TRUE)
+   test = As2[testID,]
+   train = As2[-testID,]
+
+   knn.pred.test = knn(train[,1:2], test[,1:2], cl=train$y,k=15)
+   cv.est.knn = mean(knn.pred.test != test$y)
+   return(cv.est.knn)
+ })
> mean(cv.knn)
[1] 0.02701693
>
> # knn = 16
> K=10
> cv.knn = sapply(1:K, FUN=function(i){
+   testID = which(folds == i, arr.ind = TRUE)
+   test = As2[testID,]
+   train = As2[-testID,]
+
+   knn.pred.test = knn(train[,1:2], test[,1:2], cl=train$y,k=16)
+   cv.est.knn = mean(knn.pred.test != test$y)
+   return(cv.est.knn)
+ })
> mean(cv.knn)
[1] 0.022558
>
> # knn = 20
> K=10
> cv.knn = sapply(1:K, FUN=function(i){
+   testID = which(folds == i, arr.ind = TRUE)
+   test = As2[testID,]
+   train = As2[-testID,]
+
+   knn.pred.test = knn(train[,1:2], test[,1:2], cl=train$y,k=20)
+   cv.est.knn = mean(knn.pred.test != test$y)
+   return(cv.est.knn)
+ })
> mean(cv.knn)
[1] 0.02757811
```

	LM	GLM	LDA	QDA	KNN K=1	KNN K=3	KNN K=13	KNN K=15	KNN K=16	KNN K=20
Error	0.483	0.483	0.483	0.055	0	0.01	0.015	0.014	0.014	0.021
10fold MSE	0.2507	0.2505	0.4916	0.0651	0.0274	0.03	**0.0194**	0.027	0.0225	0.0275

About KNN, I pick k = 1, 3, 13, 15, 16 and 20 to do the 10-fold cross validation. The reason is that except k=20, these errors are lower than others. Firstly, we can see. According to the table we know that QDA and KNN perform better than LM, GLM and LDA. This situation is similar to the textbook chapter 4.5 scenario 5 that the data with varying covariance and quadradic analysis boundary. After 10-fold correlation variation. We can see that the errors of LM and GLM go down but others are slightly increased. About KNN, I thought that the difference between k equals 1 to 20 are not significantly, because the variables are well classified which we can see through the plot figure. In conclusion, QDA and KNN model are fitted this data.