Tanmay Gupta
tg289@njit.edu

Homework-3

1) We now examine the differences between LDA and QDA.

(a) If the Bayes decision boundary is linear, do we expect LDA or QDA to perform better on the training set? On the test set?

Answer=>If we are working on the training data set , QDA will outperform LDA because of the decrease in bias, but on testing set if the decision boundary is linear then LDA will perform better.

(b) If the Bayes decision boundary is non-linear, do we expect LDA or QDA to perform better on the training set? On the test set?

Answer=> On both the sets whether it be training or testing, if the decision boundary is non- linear QDA will perform better than LDA.

(c) In general, as the sample size n increases, do we expect the test prediction accuracy of QDA relative to LDA to improve, decline, or be unchanged? Why?

Answer=> As the sample size increases it is expected that the prediction accuracy of QDA relative to LDA will improve. In general, this will be the case with non-linear methods relative to linear methods since with a increase in sample size there will be an decrease in bias that comes from a non-linear method, but variance will also tend to decrease as n increases.

(d) True or False: Even if the Bayes decision boundary for a given problem is linear, we will probably achieve a superior test error rate using QDA rather than LDA because QDA is flexible enough to model a linear decision boundary. Justify your answer.

Answer=> False,For small values of n, QDA will tend to model noise and thus the test error will be higher in this case.

2) Suppose we collect data for a group of students in a statistics class with variables X1 =hours studied, X2 =undergrad GPA, and Y = receive an A. We fit a logistic regression and produce estimated coefficient, ˆ β0 = −6, ˆβ1 = 0.05, ˆβ2 = 1.

(a) Estimate the probability that a student who studies for 40 h and has an undergrad GPA of 3.5 gets an A in the class.

Answer=> $P(x) = \frac{\exp(B0+B1(x1)+B2(x2)}{1+\exp(B0+B1(x1)+B2(x2)}$

$= \frac{\exp(-6+.05*40+1*3.5)}{1+ \exp(-6+.05*40+1*3.5)}$

$= \frac{\exp(-.50)}{1+ \exp(-.50)}$ = .3775 which implies 37.75%

(b) How many hours would the student in part (a) need to study to have a 50% chance of getting an A in the class?

Answer=>Let Y^=β0^+β1^x1+β2^x2=−6+0.05x1+3.5Y^=β0^+β1^x1+β2^x2=−6+0.05x1+3.5, then the logistic regression function can be expressed as p1(x)=eY^1+eY^p1(x)=eY^1+eY^ where if the probability the students gets an A in the class is .5 then, p2(x)=eY^1+eY^=.5p2(x)=eY^1+eY^=.5
so,

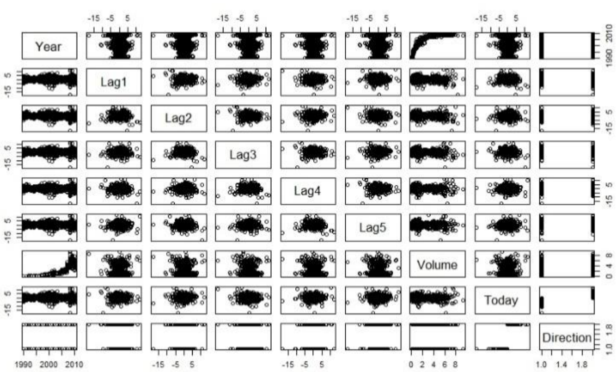
eY^=.5(1+eY^)=.5+.5eY^eY^=.5(1+eY^)=.5+.5eY^
eY^−.5eY^=.5eY^−.5eY^=.5
eY^(1−.5)=.5eY^(1−.5)=.5
.5eY^=.5.5eY^=.5
log(.5)+Y^=log(.5)log(.5)+Y^=log(.5)
Y^=−6+0.05x1+3.5=0Y^=−6+0.05x1+3.5=0
x1=50

3) This question should be answered using the Weekly data set, which is part of the ISLR package. This data is similar in nature to the Smarket data from this chapter's lab, except that it contains 1, 089 weekly returns for 21 years, from the beginning of 1990 to the end of 2010.

(a) Produce some numerical and graphical summaries of the Weekly data. Do there appear to be any patterns?

```
> library("ISLR")
> summary(Weekly)
      Year           Lag1                 Lag2                 Lag3                 Lag
4           Lag5
 Min.   :1990   Min.   :-18.1950   Min.   :-18.1950   Min.   :-18.1950   Min.   :
-18.1950   Min.   :-18.1950
 1st Qu.:1995   1st Qu.: -1.1540   1st Qu.: -1.1540   1st Qu.: -1.1580   1st Qu.:
-1.1580   1st Qu.: -1.1660
 Median :2000   Median :  0.2410   Median :  0.2410   Median :  0.2410   Median :
0.2380   Median :  0.2340
 Mean   :2000   Mean   :  0.1506   Mean   :  0.1511   Mean   :  0.1472   Mean   :
0.1458   Mean   :  0.1399
 3rd Qu.:2005   3rd Qu.:  1.4050   3rd Qu.:  1.4090   3rd Qu.:  1.4090   3rd Qu.:
1.4090   3rd Qu.:  1.4050
 Max.   :2010   Max.   : 12.0260   Max.   : 12.0260   Max.   : 12.0260   Max.   :
12.0260   Max.   : 12.0260
     Volume            Today          Direction
 Min.   :0.08747   Min.   :-18.1950   Down:484
 1st Qu.:0.33202   1st Qu.: -1.1540   Up  :605
 Median :1.00268   Median :  0.2410
 Mean   :1.57462   Mean   :  0.1499
 3rd Qu.:2.05373   3rd Qu.:  1.4050
 Max.   :9.32821   Max.   : 12.0260
```

```
> plot(Weekly)
```



As previously observed, the Lag variables are weakly correlated and there is a strong correlation between Year and Volume.

```
> cor(Weekly[,-9])
               Year         Lag1        Lag2        Lag3         Lag4         Lag5
Volume        Today
Year     1.00000000 -0.032289274 -0.03339001 -0.03000649 -0.031127923 -0.030519101
0.84194162 -0.032459894
Lag1    -0.03228927  1.000000000 -0.07485305  0.05863568 -0.071273876 -0.008183096
-0.06495131 -0.075031842
Lag2    -0.03339001 -0.074853051  1.00000000 -0.07572091  0.058381535 -0.072499482
-0.08551314  0.059166717
Lag3    -0.03000649  0.058635682 -0.07572091  1.00000000 -0.075395865  0.060657175
-0.06928771 -0.071243639
Lag4    -0.03112792 -0.071273876  0.05838153 -0.07539587  1.000000000 -0.075675027
-0.06107462 -0.007825873
Lag5    -0.03051910 -0.008183096 -0.07249948  0.06065717 -0.075675027  1.000000000
-0.05851741  0.011012698
Volume   0.84194162 -0.064951313 -0.08551314 -0.06928771 -0.061074617 -0.058517414
1.00000000 -0.033077783
Today   -0.03245989 -0.075031842  0.05916672 -0.07124364 -0.007825873  0.011012698
-0.03307778  1.000000000
```

About 44% of the data is classified as Down, and 55% is classified as Up.

```
> table(Weekly$Direction)/sum(table(Weekly$Direction))
```

```
     Down        Up
0.4444444 0.5555556
```
It is important to note that Directional is simply a nominal form of the Today feature. This should be the case since Today gives the percentage increase, or decrease of the current week, and the variable Direction simply maps these values to 'Up', and 'Down' respectively.

(b) Use the full data set to perform a logistic regression with Direction as the response and the five lag variables plus Volume as predictors. Use the summary function to print the results. Do any of the predictors appear to be statistically significant? If so,
which ones?

```
> glm.fit <- glm(Direction~.-Year-Today,data=Weekly,family="binomial")
> summary(glm.fit)

Call:
glm(formula = Direction ~ . - Year - Today, family = "binomial",
    data = Weekly)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.6949  -1.2565   0.9913   1.0849   1.4579

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  0.26686    0.08593   3.106   0.0019 **
Lag1        -0.04127    0.02641  -1.563   0.1181
Lag2         0.05844    0.02686   2.175   0.0296 *
Lag3        -0.01606    0.02666  -0.602   0.5469
Lag4        -0.02779    0.02646  -1.050   0.2937
Lag5        -0.01447    0.02638  -0.549   0.5833
Volume      -0.02274    0.03690  -0.616   0.5377
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1496.2  on 1088  degrees of freedom
Residual deviance: 1486.4  on 1082  degrees of freedom
AIC: 1500.4

Number of Fisher Scoring iterations: 4
```

(c) Compute the confusion matrix and overall fraction of correct predictions. Explain what the confusion matrix is telling you about the types of mistakes made by logistic regression.

```
> glm.probs <- predict(glm.fit,type = "response")
> glm.pred <- rep("Down",nrow(Weekly))
> glm.pred[glm.probs>0.5] = "Up"
>
> table(glm.pred,Weekly$Direction)

glm.pred Down  Up
    Down   54  48
    Up    430 557
> mean(glm.pred == Weekly$Direction)
[1] 0.5610652
> 558/(558+47)
[1] 0.922314
> 56/(428+56)
[1] 0.1157025
```

(d) Now fit the logistic regression model using a training data period from 1990 to 2008, with Lag2 as the only predictor. Compute the confusion matrix and the overall fraction of correct predictions for the held out data (that is, the data from 2009 and 2010).

```
> train <- Weekly[,"Year"] <= 2008
>
```

```
> glm.fit <- glm(Direction~Lag2,data = Weekly,subset = train, family = "binomial"
)
> summary(glm.fit)

Call:
glm(formula = Direction ~ Lag2, family = "binomial", data = Weekly,
    subset = train)

Deviance Residuals:
   Min      1Q  Median      3Q     Max
-1.536  -1.264   1.021   1.091   1.368

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  0.20326    0.06428   3.162  0.00157 **
Lag2         0.05810    0.02870   2.024  0.04298 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1354.7  on 984  degrees of freedom
Residual deviance: 1350.5  on 983  degrees of freedom
AIC: 1354.5

Number of Fisher Scoring iterations: 4


> glm.probs <- predict(glm.fit,Weekly[!train,],type = "response")
>
> glm.pred <- rep("Down",nrow(Weekly))
> glm.pred[glm.probs>0.5] = "Up"
>
> table(glm.pred,Weekly[,"Direction"]) # Confusion Matrix.

glm.pred Down  Up
    Down   63  85
    Up    421 520
> mean(glm.pred == Weekly[,"Direction"]) # Fraction of correct predictions.
[1] 0.5353535
```

(e) Repeat (d) using LDA.

```
> library(MASS)
> lda.fit <- lda(Direction~Lag2,data=Weekly,subset=train)
> lda.fit
Call:
lda(Direction ~ Lag2, data = Weekly, subset = train)

Prior probabilities of groups:
     Down        Up
0.4477157 0.5522843

Group means:
            Lag2
Down -0.03568254
Up    0.26036581

Coefficients of linear discriminants:
           LD1
Lag2 0.4414162
> lda.pred <- predict(lda.fit,Weekly[!train,])
> lda.class <- lda.pred$class
> table(lda.class,Weekly[!train,9])

lda.class Down Up
     Down    9  5
     Up     34 56
> mean(lda.class == Weekly[!train,9])
[1] 0.625
```

(f) Repeat (d) using QDA.

```
> qda.fit <- qda(Direction~Lag2,data=Weekly,subset=train)
> qda.fit
Call:
qda(Direction ~ Lag2, data = Weekly, subset = train)

Prior probabilities of groups:
     Down        Up 
0.4477157 0.5522843 

Group means:
            Lag2
Down -0.03568254
Up    0.26036581
> qda.pred <- predict(qda.fit,Weekly[!train,])
> qda.class <- qda.pred$class
> 
> table(qda.class,Weekly[!train,9])

qda.class Down Up
     Down    0  0
     Up     43 61
> mean(qda.class == Weekly[!train,9])
[1] 0.5865385
```

(g) Repeat (d) using KNN with K = 1.

```
> library(class)
> 
> train.X <- cbind(Weekly[train,3])
> test.X <- cbind(Weekly[!train,3])
> 
> train.Direction <- Weekly[train,c(9)]
> test.Direction <- Weekly[!train,c(9)]
> 
> knn.pred <- knn(train.X,test.X,train.Direction,k=1)
> 
> table(knn.pred,test.Direction)
        test.Direction
knn.pred Down Up
    Down   21 30
    Up     22 31
> mean(knn.pred == test.Direction)
[1] 0.5
```

(h) Which of these methods appears to provide the best results on
this data?

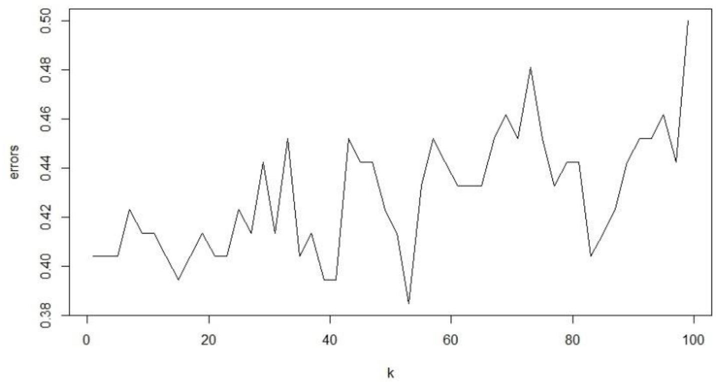
Answer=> The best performing model is LDA, since it has the highest prediction accuracy (about 68%).

(i) Experiment with different combinations of predictors, including possible transformations and interactions, for each of the methods. Report the variables, method, and associated confusion matrix that appears to provide the best results on the held out data. Note that you should also experiment with values for K in the KNN classifier.

```
> glm.fit <- glm(Direction~Lag2,data = Weekly,subset = train, family = "binomial"
)
> summary(glm.fit)

Call:
glm(formula = Direction ~ Lag2, family = "binomial", data = Weekly, 
    subset = train)

Deviance Residuals: 
   Min      1Q  Median      3Q     Max  
-1.536  -1.264   1.021   1.091   1.368  

Coefficients:
            Estimate Std. Error z value Pr(>|z|)   
(Intercept)  0.20326    0.06428   3.162  0.00157 **
Lag2         0.05810    0.02870   2.024  0.04298 * 
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

```
(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1354.7  on 984  degrees of freedom
Residual deviance: 1350.5  on 983  degrees of freedom
AIC: 1354.5

Number of Fisher Scoring iterations: 4

> glm.probs <- predict(glm.fit,Weekly[!train,],type = "response")
>
> glm.pred <- rep("Down",nrow(Weekly))
> glm.pred[glm.probs>0.55] = "Up"
>
> table(glm.pred,Weekly[,"Direction"]) # Confusion Matrix.

glm.pred Down  Up
    Down  196 266
    Up    288 339
> mean(glm.pred == Weekly[,"Direction"]) # Fraction of correct predictions.
[1] 0.4912764
> # Non Linear Transformation
> # Up to Cubic power
> glm.fit <- glm(Direction~Lag2+I(Lag2^2)+I(Lag2^3),data = Weekly,subset = train,
family = "binomial")
> summary(glm.fit)

Call:
glm(formula = Direction ~ Lag2 + I(Lag2^2) + I(Lag2^3), family = "binomial",
    data = Weekly, subset = train)

Deviance Residuals:
   Min      1Q  Median      3Q     Max
-2.194  -1.245   1.008   1.108   1.142

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) 0.1608285  0.0714552   2.251   0.0244 *
Lag2        0.0491970  0.0340597   1.444   0.1486
I(Lag2^2)   0.0095243  0.0072076   1.321   0.1864
I(Lag2^3)   0.0005092  0.0005445   0.935   0.3497
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1354.7  on 984  degrees of freedom
Residual deviance: 1348.5  on 981  degrees of freedom
AIC: 1356.5

Number of Fisher Scoring iterations: 4

> glm.probs <- predict(glm.fit,Weekly[!train,],type = "response")
>
> glm.pred <- rep("Down",nrow(Weekly))
> glm.pred[glm.probs>0.5] = "Up"
>
> table(glm.pred,Weekly[,"Direction"]) # Confusion Matrix.

glm.pred Down  Up
      Up  484 605
> mean(glm.pred == Weekly[,"Direction"]) # Fraction of correct predictions.
[1] 0.5555556
> glm.fit <- glm(Direction~sqrt(abs(Lag2)),data = Weekly,subset = train, family =
"binomial")
> summary(glm.fit)

Call:
glm(formula = Direction ~ sqrt(abs(Lag2)), family = "binomial",
    data = Weekly, subset = train)

Deviance Residuals:
   Min      1Q  Median      3Q     Max
```

```
-1.405  -1.263   1.058   1.093   1.136

Coefficients:
                Estimate Std. Error z value Pr(>|z|)
(Intercept)      0.09488    0.15028   0.631    0.528
sqrt(abs(Lag2))  0.09961    0.11788   0.845    0.398

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1354.7  on 984  degrees of freedom
Residual deviance: 1354.0  on 983  degrees of freedom
AIC: 1358

Number of Fisher Scoring iterations: 3

> # Log
> glm.fit <- glm(Direction~log10(abs(Lag2)),data = Weekly,subset = train, family
= "binomial")
> summary(glm.fit)

Call:
glm(formula = Direction ~ log10(abs(Lag2)), family = "binomial",
    data = Weekly, subset = train)

Deviance Residuals:
   Min      1Q  Median      3Q     Max
-1.305  -1.269   1.074   1.088   1.167

Coefficients:
                 Estimate Std. Error z value Pr(>|z|)
(Intercept)       0.20916    0.06410   3.263   0.0011 **
log10(abs(Lag2))  0.06823    0.13149   0.519   0.6038
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1354.7  on 984  degrees of freedom
Residual deviance: 1354.4  on 983  degrees of freedom
AIC: 1358.4

Number of Fisher Scoring iterations: 3

> train.X <- cbind(Weekly[train,3])
> test.X <- cbind(Weekly[!train,3])
>
> train.Direction <- Weekly[train,c(9)]
> test.Direction <- Weekly[!train,c(9)]
>
>
> errors <- c()
>
> maxK <- 100
> step <- 2
>
> for(j in seq(1,maxK,step)){
+   knn.pred <- knn(train.X,test.X,train.Direction,k=j)
+   table(knn.pred,test.Direction)
+   errors <- c(1-mean(knn.pred == test.Direction),errors)
+ }
>
> data <- cbind(seq(1,maxK,step),errors)
> plot(data,type="l",xlab="k")
```



for KNN-

```
> knn.pred <- knn(train.X,test.X,train.Direction,k=which.min(errors))
> table(knn.pred,test.Direction)
        test.Direction
knn.pred Down Up
    Down   17 24
    Up     26 37
> mean(knn.pred == test.Direction)
[1] 0.5192308
```

4) We now review k-fold cross-validation.
(a) Explain how k-fold cross-validation is implemented.
Answer=> // k-fold
divide train data into k parts
for i = 1 to k
train network using k-1 parts
compute accuracy using 1 part
end for
compute average accuracy of the k runs

(b) What are the advantages and disadvantages of k-fold cross-validation
relative to:
i. The validation set approach?
Answer=> The main disadvantage of using an approach such as k-fold is the computational cost involved. This may provide too time consuming or costly for some models. The validation set approach has a clear benefit in this case, since the model only needs be learnt from the data once, and tested once. However, a validation set may not be available (due to only a small number of observations being available) or may be too costly to obtain in practice. In these cases, the k-fold cross validation is a clear winner, for it allows us to fine tune our model. Furthermore, a validation set may tend to over estimate the error since less data is used for training.
ii. LOOCV?
Answer=> The Leave-One-Out Cross Validation (LOOCV) approach has a worse (or the same, in the case k=n) computational cost to K-Fold Cross Validation since the model needs to be trained and tested n times, instead of k. Furthermore, LOOCV suffers from a higher variance in result, since they are typically highly correlated (most models trained are very similar). There are no significant advance to using LOOCV since 5-fold or 10-fold will significantly reduce the computational cost and will neither suffer from high bias or variance.

5) In Chapter 4, we used logistic regression to predict the probability of default using income and balance on the Default data set. We will now 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 uses income and balance to predict default.

```
> library(ISLR)
> glm.fit=glm(default~income+balance,family='binomial',data=Default)
> print(glm.fit)

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

```
    data = Default)

Coefficients:
(Intercept)       income      balance
 -1.154e+01    2.081e-05    5.647e-03

Degrees of Freedom: 9999 Total (i.e. Null);  9997 Residual
Null Deviance:      2921
Residual Deviance: 1579        AIC: 1585
```

(b) Using the validation set approach, estimate the test error of this model. In order to do this, you must perform the following steps:

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

```
> set.seed(3)
> subset=sample(1:1000,500)
```

ii. Fit a multiple logistic regression model using only the training observations.

```
> glm.fit=glm(default~income+balance,family='binomial',data=Default,subset=subset
)
> print(glm.fit)

Call:  glm(formula = default ~ income + balance, family = "binomial",
    data = Default, subset = subset)

Coefficients:
(Intercept)       income      balance
 -1.072e+01    1.306e-05    5.379e-03

Degrees of Freedom: 499 Total (i.e. Null);  497 Residual
Null Deviance:      120.6
Residual Deviance: 72.21       AIC: 78.21
```

iii. Obtain a prediction of default status for each individual in the validation set by computing the posterior probability of default for that individual, and classifying the individual to the default category if the posterior probability is greater than 0.5.

```
> glm.resp=predict(glm.fit,Default[-subset,],type='response')
> glm.pred=ifelse(glm.resp>0.5,'Yes','No')
> glm.pred
```

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

```
> mean(glm.pred!=Default[-subset,'default'])
[1] 0.02631579
```

(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.

```
> DefaultValid = function(formula=default~income+balance,n=1000,s=500,seed){
+   set.seed(seed)
+   subset=sample(n,s)
+   glm.fit=glm(formula,family='binomial',
+               data=Default,subset=subset)
+
+   glm.resp=predict(glm.fit,Default[-subset,],type='response')
+
+   mean(glm.pred!=Default[-subset,'default'])
+ }
>
> for(i in 1:3) print(DefaultValid(seed=i))
[1] 0.02715789
[1] 0.02621053
[1] 0.02631579
```

(d) Now consider a logistic regression model that predicts the probability of default using income, balance, and a dummy variable for student. Estimate the test error for this model using the validation set approach. Comment on whether or not including a dummy variable for student leads to a reduction in the test error rate.

```
> for(i in 1:3) print(DefaultValid(formula=default~income+balance+student,seed=i)
)
[1] 0.02715789
[1] 0.02621053
[1] 0.02631579
```