CLASSIFICATION: LDA and QDA

1. Chap 4, exercise 5

(a) On a training set, LDA and QDA are both expected to perform well. LDA uses a correct model, and QDA includes the linear model as a special case. QDA, including a wider range of models, will fit the given particular data set better. On a test set, however, LDA should be more accurate because QDA, with additional parameters and an overfit, will have a higher variance.
(b) For a nonlinear boundary, QDA should perform better on both the training and test sets because LDA will not capture the nonlinear pattern.
(c) As the sample size increases, QDA will perform better because its higher variance will be offset by a large sample size.
(d) False. A higher variance of QDA prediction, without any gain in bias, will result in a higher error rate.

2. Chap 4, exercise 7

Two groups of stocks are those issuing dividends (group 1) and not issuing dividends (group 2). We are given group parameters μ_1=10, μ_2=0 (estimated from data), σ^2=36, prior probabilities π_1=0.8 and π_2=0.2, the observed value x=4, and the normal distribution of data (last year's percent profit). Then, by Bayes' Theorem,

$$
\begin{aligned}
p_1(x=4) &= \boldsymbol{P}\{Y=1 \mid X=4\} \\
&= \frac{\pi_1 f_1(x=4)}{\pi_1 f_1(x=4) + \pi_2 f_2(x=4)} \\
&= \left.\frac{\pi_1 \frac{1}{\sigma\sqrt{2\pi}} \exp\{-(x-\mu_1)^2/2\sigma^2\}}{\pi_1 \frac{1}{\sigma\sqrt{2\pi}} \exp\{-(x-\mu_1)^2/2\sigma^2\} + \pi_2 \frac{1}{\sigma\sqrt{2\pi}} \exp\{-(x-\mu_2)^2/2\sigma^2\}}\right|_{x=4} \\
&= \frac{(0.8)\exp\{-(4-10)^2/72\}}{(0.8)\exp\{-(4-10)^2/72\} + (0.2)\exp\{-(4-0)^2/72\}} \\
&= \frac{0.4852}{0.4852+0.1601} = \boxed{0.7519}
\end{aligned}
$$

Although this company's last year profit is closer to μ_2 than to μ_1, group 1 has a higher prior probability. As a result, group 1 is still more likely, and the posterior probability that a company will pay dividends is **0.7519**.

3. Chap. 4, exercise 13 (Weekly data project).

(a) Get Weekly data and apply LDA, predicting Direction of the market, separating training and testing data.

```
> install.packages("ISLR2"); library(ISLR); attach(Weekly);
```

```
> Weekly_training = Weekly[ Year <= 2008, ]      # Split the data by year into
> Weekly_testing = Weekly[ Year > 2008, ]        # the training and testing sets

> library(MASS)                                  # This library contains LDA
> lda.fit = lda(Direction ~ Lag2, data=Weekly_training)
# Fit using training data only

# Now use this model to predict classification of the testing data
> Direction_Predicted_LDA = predict(lda.fit, Weekly_testing)$class

# Construct the confusion matrix
> table( Weekly_testing$Direction, Direction_Predicted_LDA )

         Direction_Predicted_LDA
Direction Down Up
     Down    9 34
     Up      5 56

> mean( Weekly_testing$Direction == Direction_Predicted_LDA )
[1] 0.625
# The correct classification rate of LDA is 62.5% (the error rate is 37.5%).


(b) # QDA
> qda.fit = qda(Direction ~ Lag2, data=Weekly_training)
> Direction_Predicted_QDA = predict(qda.fit, Weekly_testing)$class
> table( Weekly_testing$Direction, Direction_Predicted_QDA )

         Direction_Predicted_QDA
Direction Down Up
     Down    0 43
     Up      0 61

> mean( Weekly_testing$Direction == Direction_Predicted_QDA )
 [1] 0.5865385

# Correct classification rate of QDA is only 58.7%.
# However, QDA always predicted that the market goes Up.
# LDA has a higher prediction power. QDA seems to be an overfit.


# Logistic regression and LDA yield the best results so far.


(c) Among the methods that we explored on this data set,
     LDA yields the highest classification rate, 0.625.
     Logistic regression and QDA yield the classification rate 0.587.
     KNN yields the classification rate 0.570.
Hence, LDA appears to have the best prediction accuracy.
```

4. <u>Chap 4, exercise 16 (Boston data project).</u>

```
> attach(Boston)          # Attach a new dataset "Boston" from the MASS package
> ?Boston                 # Data description
> names(Boston)
 [1] "crim"   "zn"   "indus" "chas"  "nox"  "rm"  "age"  "dis"  "rad"  "tax"  "ptratio"
"black" "lstat" "medv"

> Crime_Rate = ( crim > median(crim) )      # Categorical variable Crime_Rate
> table(Crime_Rate)
     Crime_Rate
     FALSE  TRUE
       253   253                            # The median splits the data evenly

> Boston = cbind( Boston, Crime_Rate )
```

 # <u>Logistic regression</u>… First, we'll fit the full model.

```
> lreg.fit = glm(Crime_Rate ~ zn + indus + chas + nox + rm + age + dis + rad + tax
+ ptratio + black + lstat + medv, family="binomial")
> summary(lreg.fit)

              Estimate Std. Error z value Pr(>|z|)
(Intercept) -34.103704   6.530014  -5.223 1.76e-07 ***
zn           -0.079918   0.033731  -2.369  0.01782 *
indus        -0.059389   0.043722  -1.358  0.17436
chas          0.785327   0.728930   1.077  0.28132
nox          48.523782   7.396497   6.560 5.37e-11 ***
rm           -0.425596   0.701104  -0.607  0.54383
age           0.022172   0.012221   1.814  0.06963 .
dis           0.691400   0.218308   3.167  0.00154 **
rad           0.656465   0.152452   4.306 1.66e-05 ***
tax          -0.006412   0.002689  -2.385  0.01709 *
ptratio       0.368716   0.122136   3.019  0.00254 **
black        -0.013524   0.006536  -2.069  0.03853 *
lstat         0.043862   0.048981   0.895  0.37052
medv          0.167130   0.066940   2.497  0.01254 *
```

 # Some variables are not significant. We'll fit a reduced model, without them.

```
> lreg.fit = glm(Crime_Rate ~ zn + nox + age + dis + rad + tax + ptratio + black +
medv, family="binomial")

> summary(lreg.fit)

              Estimate Std. Error z value Pr(>|z|)
(Intercept) -31.441272   6.048989  -5.198 2.02e-07 ***
zn           -0.082567   0.031424  -2.628  0.00860 **
nox          43.195824   6.452812   6.694 2.17e-11 ***
age           0.022851   0.009894   2.310  0.02091 *
dis           0.634380   0.207634   3.055  0.00225 **
rad           0.718773   0.142066   5.059 4.21e-07 ***
```

```
tax          -0.007676   0.002503  -3.066  0.00217 **
ptratio       0.303502   0.109255   2.778  0.00547 **
black        -0.012866   0.006334  -2.031  0.04224 *
medv          0.112882   0.034362   3.285  0.00102 **
```

 # To evaluate the model's predictive performance, we'll use the validation set method.

```
> n = length(crim)          # Split the data randomly into training and testing subsets
> n
[1] 506
> Z = sample(n,n/2)
> Data_training = Boston[ Z, ]
> Data_testing = Boston[ -Z, ]

> lreg.fit = glm(Crime_Rate ~ zn + nox + age + dis + rad + tax + ptratio + black + medv, data=Data_training, family="binomial")
                           # Fit a logistic regression model on the training data
                           # and use this model to predict the testing data
> Predicted_probability = predict( lreg.fit, data.frame(Data_testing), type="response" )
> Predicted_Crime_Rate = 1*(Predicted_probability > 0.5)
# Predicted crime rate:  1 = high, 0 = low

> table( Data_testing$Crime_Rate,  Predicted_Crime_Rate )

   Predicted_Crime_Rate
      0   1
  0 116  11
  1  14  98

> mean( Data_testing$Crime_Rate == Predicted_Crime_Rate )
[1] 0.8953975
```

 # Logistic regression shows the correct classification rate of 89.5%, the error rate is 10.5%.

 # Now, try LDA. For a fair comparison, use the same training data, then predict the testing data.

```
> lda.fit = lda(Crime_Rate ~ zn + nox + age + dis + rad + tax + ptratio + black + medv, data=Data_training)

> Predicted_Crime_Rate_LDA = predict(lda.fit, data.frame(Data_testing))$class
> table( Data_testing$Crime_Rate, Predicted_Crime_Rate_LDA )

   Predicted_Crime_Rate_LDA
      0   1
```

```
  0 117  10
  1  28  84

> mean( Data_testing$Crime_Rate != Predicted_Crime_Rate_LDA )
[1] 0.1589958
```

 # LDA’s error rate is higher, 15.9%.

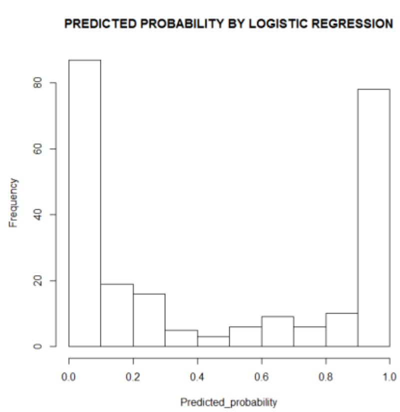
 # What about QDA? If logit has a nonlinear relation with predictors, QDA may be a better method.

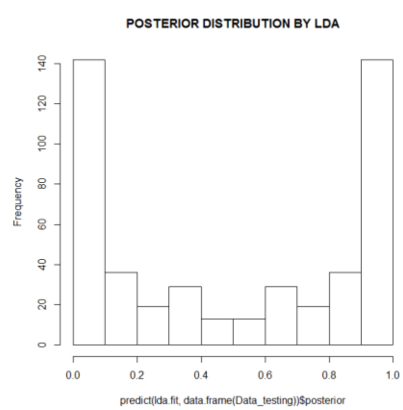
```
> qda.fit = qda(Crime_Rate ~ zn + nox + age + dis + rad + tax + ptratio + black + medv, data=Data_training)
> Predicted_Crime_Rate_QDA = predict(qda.fit, data.frame(Data_testing))$class
> table( Data_testing$Crime_Rate, Predicted_Crime_Rate_QDA )

   Predicted_Crime_Rate_QDA
      0   1
  0 121   6
  1  26  86

> mean( Data_testing$Crime_Rate != Predicted_Crime_Rate_QDA )
[1] 0.1338912
```

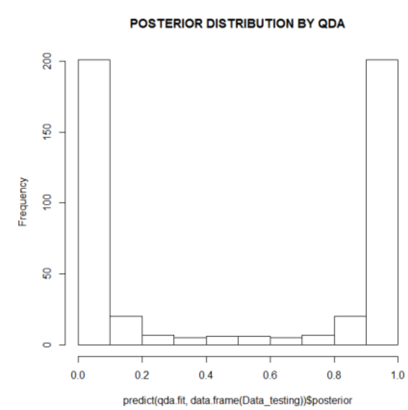
 # QDA’s error rate of 13.4% is between the LDA and the logistic regression. According to these quick results, logistic regression has the best prediction power.

 # It may be interesting to look at the distribution of predicted probabilities of high crime zones, produced by the three classification methods. All of them mostly discriminate between the two groups pretty well.

```
> hist(Predicted_probability, main='PREDICTED PROBABILITY BY LOGISTIC REGRESSION')
> hist(predict(lda.fit, data.frame(Data_testing))$posterior, main='POSTERIOR DISTRIBUTION BY LDA')
> hist(predict(qda.fit, data.frame(Data_testing))$posterior, main='POSTERIOR DISTRIBUTION BY QDA')
```





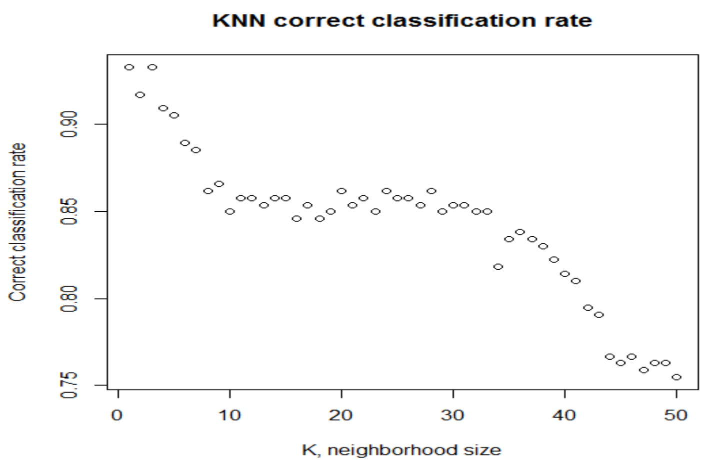


KNN method

```
# Define training and testing data
> names(Data_training)
 [1] "crim"      "zn"        "indus"     "chas"      "nox"
 [6] "rm"        "age"       "dis"       "rad"       "tax"
[11] "ptratio"   "black"     "lstat"     "medv"      "Crime_Rate"
```

 # Variables zn, nox, age, dis, rad, tax, ptratio, black, medv are in columns 2,5,7,8,9,10,11,12,14.

```
> X.train = Data_training[ , c(2,5,7,8,9,10,11,12,14) ]
> X.test = Data_testing[ , c(2,5,7,8,9,10,11,12,14) ]
> Y.train = Data_training$Crime_Rate
> Y.test = Data_testing$Crime_Rate

# The neighborhood size K is not specified, so let us try to find the best K.
> knn.rate = rep(0,50)
> for (k in 1:50){ knn.fit = knn( X.train, X.test, Y.train, k )
+                  knn.rate[k] = mean( Y.test == knn.fit )  }
> plot( 1:50, knn.rate, xlab="K, neighborhood size", ylab="Correct classification rate", main="KNN correct classification rate")
```




```
> which.max(knn.rate)
[1] 1
> knn.rate[1]
[1] 0.9328063
# The optimal size is K=1, the most flexible KNN procedure.
```

 # To summarize, here are the correct classification rates:

1. **KNN with K=1 - 93.3%**
2. **Logistic regression – 89.5%**
3. **QDA - 86.6%**
4. **LDA - 84.1%**

KNN appears to yield the best prediction accuracy, followed by logistic regression, QDA, and LDA.