Assignment 4
Shaheed Shihan
Modern Applied Statistics II

3. **This problem relates to the QDA model, in which the observations within each class are drawn from a normal distribution with a class specific mean vector and a class specific covariance matrix. We consider the simple case where p = 1; i.e. there is only one feature.**

Suppose that we have K classes, and that if an observation belongs to the kth class then X comes from a one-dimensional normal distribution, $X \sim N(\mu_k, \sigma_k^2)$. Recall that the density function for the one-dimensional normal distribution is given in (4.11). Prove that in this case, the Bayes' classifier is not linear. Argue that it is in fact quadratic.

Hint: For this problem, you should follow the arguments laid out in Section 4.4.2, but without making the assumption that $\sigma_1^2 = \cdots = \sigma_k^2$

Answer:

$$log\ p_k(x) = \log \pi_k + \log \frac{1}{\sqrt{2\pi}\sigma_k} - \frac{1}{2\sigma_k^2}(x-\mu)^2$$
$$= \log \pi_k - \log \sigma_k - log\frac{1}{\sqrt{2\pi}} - \frac{1}{2\sigma_k^2}(x^2 - 2x\mu + \mu^2)$$
$$\delta_k(x) = \log \pi_k - \log \sigma_k - \frac{x^2}{2\sigma_k^2} + \frac{x\mu}{\sigma_k^2} - \frac{\mu^2}{2\sigma_k^2}$$

It is quadratic because we have an x^2term.

5. 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?
On the training set, the QDA has a chance of performing better because of its flexibility but on the test set, LDA should by definition, 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?
We can expect the QDA to perform better on both.

(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?
As sample size increases the constant variance assumption of LDA might get violated. Therefore since QDA has more flexibility, the test prediction accuracy of QDA relative to LDA might improve.

(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.
False. QDA is flexible and might work better on the training set, but in the test set, this flexibility might translate to overfitting.

10. e.

```
> ldafit
Call:
lda(Direction ~ Lag2, data = Weekly, family = binomial, 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

> table(lda.pred$class, Direction.2008)
      Direction.2008
       Down Up
  Down    9  5
  Up     34 56
> mean(lda.pred$class==Direction.2008)
[1] 0.625
```

The model predicted the direction correctly 62.5% of the time. Also, when the market is on a up trend, the model is correct about 91.8% of the time.

f.

```
> 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

> table(qda.pred$class, Direction.2008)
      Direction.2008
       Down Up
  Down    1  1
  Up     42 60
> mean(qda.pred$class==Direction.2008)
[1] 0.5865385
```

We can say that the model is correct about 58.6% of the time. On an up-trend it is accurate 98% of the time – which is great until you consider the fact that when the market is going down, it is only correct about 1% of the time.

g.

```
> table(knn.pred, Direction.2008)
        Direction.2008
knn.pred Down Up
    Down   21 30
    Up     22 31
> (21+31)/(21+22+30+31)
[1] 0.5
```

h. Among the three, LDA appears to give the best model. LDA predicted accurately almost 62.5% of the time as compared to QDA which had a rate of 58.65% and KNN which was 50% (no better than a coin toss).

i.

- Varying K in the KNN model:

```
knn.pred <- knn(train.X, test.X, train.Direction, k=10)
table(knn.pred, Direction.2008)
        Direction.2008
knn.pred Down Up
    Down   19 21
    Up     24 40
> mean(knn.pred != Direction.2008)
[1] 0.4326923
```

```
> knn.pred <- knn(train.X, test.X, train.Direction, k=100)
> table(knn.pred, Direction.2008)
        Direction.2008
knn.pred Down Up
    Down   10 11
    Up     33 50
> mean(knn.pred != Direction.2008)
[1] 0.4230769

> knn.pred <- knn(train.X, test.X, train.Direction, k=150)
> table(knn.pred, Direction.2008)
        Direction.2008
knn.pred Down Up
    Down    9  5
    Up     34 56
> mean(knn.pred != Direction.2008)
[1] 0.375


> knn.pred <- knn(train.X, test.X, train.Direction, k=200)
> table(knn.pred, Direction.2008)
        Direction.2008
knn.pred Down Up
    Down    2  1
    Up     41 60
> mean(knn.pred != Direction.2008)
[1] 0.4038462
```

K=150 yielded a model with the lowest error rate.

- Changing the Lag to include other variables didn't yield a better model for the QDA.

```
> qda.fit <- qda(Direction ~ Lag2:Lag1, data = Weekly, subset = train)
> table(qda.pred$class, Direction.2008)
      Direction.2008
       Down Up
  Down   16 32
  Up     27 29
> mean(qda.pred$class==Direction.2008)
[1] 0.4326923
```

```
> qda.fit <- qda(Direction ~ Lag1:Lag5, data = Weekly, subset = train)
> table(qda.pred$class, Direction.2008)
      Direction.2008
       Down Up
  Down   10 14
  Up     33 47
> mean(qda.pred$class==Direction.2008)
[1] 0.5480769

> qda.fit <- qda(Direction ~ Lag2 +Volume, data = Weekly, subset = train)
> qda.pred <- predict(qda.fit, Weekly.2008)
> table(qda.pred$class, Direction.2008)
      Direction.2008
       Down Up
  Down   32 44
  Up     11 17
> mean(qda.pred$class==Direction.2008)
[1] 0.4711538
```

- Changing the predictor variables in LDA decreased the accuracy rate.

```
> ldafit <- lda(Direction ~ Lag2:Lag1, data = Weekly, subset = train, family = binomial)
> lda.pred <- predict(ldafit, Weekly.2008)
> table(lda.pred$class, Direction.2008)
      Direction.2008
       Down Up
  Down    0  1
  Up     43 60
> mean(lda.pred$class==Direction.2008)
[1] 0.5769231

> ldafit <- lda(Direction ~ Lag2 + Volume, data = Weekly, subset = train)
> lda.pred <- predict(ldafit, Weekly.2008)
> table(lda.pred$class, Direction.2008)
      Direction.2008
       Down Up
  Down   20 25
  Up     23 36
> mean(lda.pred$class==Direction.2008)
[1] 0.5384615

> ldafit <- lda(Direction ~ Lag1:Lag5 + Volume, data = Weekly, subset = train)
> lda.pred <- predict(ldafit, Weekly.2008)
> table(lda.pred$class, Direction.2008)
      Direction.2008
       Down Up
  Down   31 39
  Up     12 22
> mean(lda.pred$class==Direction.2008)
[1] 0.5096154
```

11. d.

```
> table(lda.pred2$class, A.test$mpg01)

     0   1
  0 127   6
  1  22 139
> mean(lda.pred2$class !=A.test$mpg01)
[1] 0.0952381
```

We have a test error rate of 9.5%.

e.

```
> table(qda.pred2$class, A.test$mpg01)

     0   1
  0 133  10
  1  16 135
> mean(qda.pred2$class !=A.test$mpg01)
[1] 0.08843537
```

We obtain an even lower error rate (8.8%) as compared to the LDA.

g.

```
> knn.pred2 <- knn(train.X2, test.X2, train.mpg01, k=1)
> table(knn.pred2, test.mpg01)
         test.mpg01
knn.pred2   0   1
        0 128  23
        1  21 122
> mean(knn.pred2 != test.mpg01)
[1] 0.1496599
```

```
> knn.pred2 <- knn(train.X2, test.X2, train.mpg01, k=5)
> table(knn.pred2, test.mpg01)
         test.mpg01
knn.pred2   0   1
        0 136  26
        1  13 119
> mean(knn.pred2 != test.mpg01)
[1] 0.1326531

> knn.pred2 <- knn(train.X2, test.X2, train.mpg01, k=10)
> table(knn.pred2, test.mpg01)
         test.mpg01
knn.pred2   0   1
        0 136  18
        1  13 127
> mean(knn.pred2 != test.mpg01)
[1] 0.1054422

> knn.pred2 <- knn(train.X2, test.X2, train.mpg01, k=15)
> table(knn.pred2, test.mpg01)
         test.mpg01
knn.pred2   0   1
        0 132  17
        1  17 128
> mean(knn.pred2 != test.mpg01)
[1] 0.1156463
```

The lowest error rate for the KNN model was at K=10. Once we got to K=15, the error rate started to increase again. Any value of K from 7 through 10 seemed to work fine.

5.
The paper “Statistical Classification Methods in Consumer Credit Scoring: A Review” examines particular problems in the credit scoring context and review the statistical models that are applied. The paper deals with a classic classification problem: whether to classify applicants of credit into a ‘good’ or ‘bad’ risk class. In recent times, this has seen more significance because of the sheer volume of growth in consumer credit.

The output variable is credit scoring – a term that is used to describe the more formal process of how likely applicants are to default with their repayments. In the banking world, statistical models called score-cards or classifiers and use predictor variables on the application forms to find the estimates of the probabilities of defaulting. Predictor variables are usually referred to as characteristics and the values of these characteristics are called attributes.

Although, minimizing risk is one goal of the credit scoring effort in the industry, there are other factors that affect the overall credit granting decision. Maximizing profits is one of them. Low-risk applicants who make regular payments on their credit cards means that financial institutions cannot charge them interest. On the other hand, high risk applicants can be profitable due to the high interest rate that can be charged on their balance, thus maximizing profits.

The paper talks about the historical statistical methods used in classifying customers into good bad risk pools. Discriminant analysis, linear regression, logistic regression and decision trees are the most common statistical methods used in the industry. Neural Networks and non-parametric methods such as K-nearest neighbor methods have also been applied by some experts.

The paper states that it is difficult to pick one model over the other. In terms of the easiness to comprehend, regression, KNN and tree-based methods are more user friendly and therefore more common approaches as compared to Neural Networks. Neural networks aren’t perfect either. While they are very well suited to model situations where we have a poor understanding of the data structure, the banking industry has been constructing score-cards on similar data for decades, and thus Neural networks haven’t been adopted either. Needless to say, Neural networks take much longer and more processing power to train.

The paper concludes with some challenges that might be of interest to statisticians. Loan servicing and review functions, risk-based pricing, fraud, profitability scoring, delinquent loan are all areas that can be improved. The problems are definitely statistical in nature and require sophisticated models than in practice at the time this paper was written.

6.
The data set that I found interesting was Student Performance Data Set on Portuguese students in the subjects of Math and Portuguese. The data set characteristics were multivariate and it was both a classification and regression problem. There were 649 entries – a small data set but probably enough to find correlations between the 30 predictor variables and the output variables which included first, second and the final grade. The problem was to correctly predict the first and second grade using only the final grade. It was easier to predict the final grade since that was dependent on the first two. If students did decently on the first and second exam they tended to do well on the final. However, the challenge was to use the final grades to predict the other two, since such prediction was more useful.
I delved into the research paper a bit more. Besides intuitive factors that affected grades such as past evaluations, other factors such as alcohol consumption seemed to have high influence on grades.
Neural Networks, Support Vector Machines, Decision Trees and Random Forests were all used on the data set. The data set was also divided for those with and without previous grades. This was essentially the testing and training set.
The data mining goals were:

- Binary classification (pass/fail)
- Classification with give levels (from I which was very good to V –insufficient)
- Regression with a numeric output for scores that ranged from 0 (0%) and 20 (100%).