STAT202 Homework 3 Solution

Problem 1
(a) Since X is uniformly distributed on [0, 1] ,the probability that a given observation will be used to make the prediction is 10%. Thus, when p = 1, on average 10% of the available observations will be used to make the prediction.

(b)) When p = 2, the probability that a given observation will be used to make the prediction is $0.1 \times 0.1 = 0.01$. Thus, on average, 1% of the available observations will be used to make the prediction.

(c) For the same reason, When $p = 100$, on average, a fraction of 0.1^p of the available observations will be used to make the prediction.

(d) When p is large, 0.1^p will be very small. So there are very few training observations near any given test observation.

(e) For p = 1, the length of each side of the hypercube is 0.1^1. For p=2, the length of each side of the hypercube is $0.1^{\frac{1}{2}}$. For $p = 100$, the length of each side of the hypercube is $0.1^{\frac{1}{100}}$. Generally, for p dimensional cases, the length of each side of the hypercube is $0.1^{\frac{1}{p}}$. When p is large, the length of each side will be near to 1. This means that it is very rare that a random sample is close in every dimension to a test point, in consequence, a non-parametric approach like K-nearest neighbors often performs poorly when p is large.

Figure 1 illustrates the phenomenon. In figure 1, black points are 100 random samples in unit square, i.e. for each point $x_1 \sim \text{Unif}(0,1)$, $x_2 \sim \text{Unif}(0,1)$. In the two red squares of side length 0.1, there are either 2 or 1 points. Vertical lines on x-axis are x_1 values for all sample points, they are also $\text{Unif}(0,1)$ distributed, in the two regions of length 0.1, there are 5 and 9 samples respectively.

Problem 2
A more flexible model usually fits training data better than a less flexible model. Performance on test data can be considered by bias-variance tradeoff.

Assuming given class label, data is from Gaussian mixture, then

(a) If the Bayes decision boundary is linear, we would expect LDA to outperform QDA on test set because it has both smaller bias and variance.

(b) If the Bayes decision boundary is non-linear, we would expect QDA to outperform LDA in general because it has smaller bias. It may not outperform LDA if sample size of training data is small or dimension is large, in which case QDA has larger variance.

(c) As sample size increases, in general we would expect test prediction accuracy of QDA would improve relative to LDA because its variance would decrease.

(d) While it is true that QDA is flexible enough to model a linear decision boundary, it may not have a smaller test error rate if the sample size is small or the dimension of the problem is large, in both cases QDA would have a large variance.

Problem 3
(a) The required probability is

$$\Pr(Y|X) = \frac{\exp(\beta_0 + \beta_1 * X_1 + \beta_2 * X_2)}{1 + \exp(\beta_0 + \beta_1 * X_1 + \beta_2 * X_2)} = 0.3775$$

b) Suppose the student needs x_1 hours to have a 50% chance of getting an A in the class. Then,

$$\frac{\exp(-6 + 0.05 * X_1 + 3.5)}{1 + \exp(-6 + 0.05 * X_1 + 3.5)} = 0.5$$

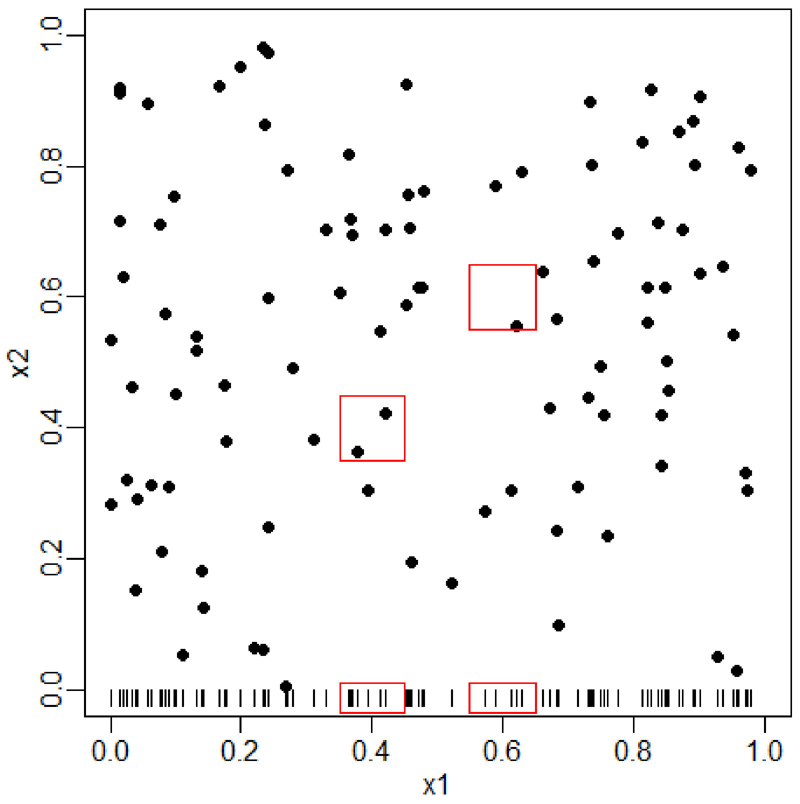

Figure 1: *Curse of dimensionality* Black points are 100 samples uniformly distributed in unit square. In the two red squares whose side length is 0.1, there are 2 and 1 samples respectively. x_1 of all random points are also uniformly distribued in $(0, 1)$. In the two regions of length 0.1, there are 5 and 9 points respectively (see x_1 axis).

Solving this equality, we get $x_1 = 50$. So the student needs 50 hours to have a 50% chance of getting an A in the class.

Problem 4
Notice that for 1-nearest neighbors, the error on the training dataset would be exactly 0. Thus the error on the test set is 36% for 1-nearest neighbors, which is higher than the logistic regression. Our suggestion should be based on the performance on the test set. So we prefer to use logistic regression for classification of new observations.

Problem 5
(a) The correlations between the lag variables and todays returns are close to zero. In other words, there appears to be little correlation between todays returns and previous days returns. The only substantial correlation is between Year and Volume. By plotting the data we see that Volume is increasing over time. In other words, the average number of shares traded daily generally increased from 1990 to 2010.

```
cor(Weekly[,-9]) #correlation between today and lags are weak
pairs(Weekly)
```

(b) Only Lag2 appears to be statistically significant, its estimated value is 0.058, and standard error is 0.027.

```
glm.full<-glm(Direction~Lag1+Lag2+Lag3+Lag4+Lag5+Volume,data=Weekly,family=binomial)
summary(glm.full) #produces summary of logistic regression
```

(c) Our model correctly predicted that the market would go up on 557 weeks and that it would go down on 54 weeks. In this case, logistic regression correctly predicted the movement of the market 56.1% of the time. We can see that the logistic regression predicts a lot of up when the Direction is actually down.

```
glm.probs<-predict(glm.full,type="response") #prediction using fitted model
glm.class<-ifelse(glm.probs>0.5,"Up","Down") #convert probability into class label

table.glm<-table(glm.class,Weekly$Direction) #confusion table
glm.class Down  Up
     Down   54  48
     Up    430 557

mean(glm.class==Weekly$Direction) #fraction of correct predictions
[1] 0.5610652
```

(d) The overall fraction of correct predictions for the held out data is 62.5%.

```
training.ind<-Weekly$Year<=2008
#fit glm model
glm.fit<-glm(Direction~Lag2,data=Weekly,subset=training.ind,family=binomial)
#prediction
glm.probs<-predict(glm.fit,Weekly[!training.ind,],type="response")
glm.class<-ifelse(glm.probs>0.5,"Up","Down")

table.glm<-table(glm.class,Weekly[!training.ind,]$Direction)#confusion table
glm.class Down Up
     Down    9  5
     Up     34 56

mean(glm.class==Weekly[!training.ind,]$Direction) #fraction of correct predictions in test data
[1] 0.625
```

(e) The overall fraction of correct predictions for the held out data is 62.5%.

```
library(MASS)
#fit LDA
lda.fit<-lda(Direction~Lag2,data=Weekly,subset = training.ind)
#predict with LDA
lda.pred<-predict(lda.fit,Weekly[!training.ind,])
#test with LDA
lda.class<-lda.pred$class

table.lda<-table(lda.class,Weekly[!training.ind,]$Direction) #confusion table
lda.class Down Up
     Down    9  5
     Up     34 56

mean(lda.class==Weekly[!training.ind,]$Direction) #fraction of correct predictions in test data
[1] 0.625
```

(f) The overall fraction of correct predictions for the held out data is 58.7%. The code is not shown, replace "lda" to "qda" would suffice. Confusion table and correct prediction rate are

```
qda.class Down Up
     Down    0  0
     Up     43 61
mean(qda.class==Weekly[!training.ind,]$Direction)
[1] 0.5865385
```

(g) The overall fraction of correct predictions for the held out data is 50.0%.

```
knn.pred<-knn(as.data.frame(Weekly[training.ind,]$Lag2),
              as.data.frame(Weekly[!training.ind,]$Lag2),Weekly[training.ind,]$Direction,k=1)
table.knn<-table(knn.pred,Weekly[!training.ind,]$Direction)
knn.pred Down Up
    Down   21 30
    Up     22 31
mean(knn.pred==Weekly[!training.ind,]$Direction)
[1] 0.5
```

(h) Logistic regression and LDA give equally best results on this data.

(i) Several interaction variables could be important in this dataset. For example, if the Lag variables are consistently positive or negative (positive interaction variable), that could be indicative of a trend in the market. It may also be important to weight each lag variable by the volume of trading on a given day, and that could be accomplished by including interaction variables between Volume and Lag1 through Lag5. We will consider the following variables: We will implement a forward selection scheme for LDA, which greedily minimizes the test error as it adds variables to the model one at a time. Several models achieve a fraction of correct predictions in the test set of 65.4%. However, the improvement in performance compared to LDA using the Lag2 variable exclusively isnt large. In addition, we are comparing many models through their performance on the test set, which amounts to fitting to the test set. As we will see in future lectures, we can actually overfit to the test set in this way, hurting the performance of model selection.

Problem 6

(a) The following code creates a binary variable *mpg01*, which contains a 1 if *mpg* value is above its median and 0 otherwise.

```
mpg01<-ifelse(mpg>median(auto$mpg),1,0)
```

We then combine it with other variables in the dataset to create a new data frame:

```
Auto<-data.frame(cbind(mpg01,auto[,-1]))
```

(b) See solution 2 for scatterplot between *mpg* and other variables. Figure 2 plots *mpg* versus log(*displacement*), and boxplot of *mpg01* versus log(*displacement*).*mpg* is negatively correlated with horsepower, displacement, weight, positively correlated with acceleration and year.Horsepower, displacement and weight are strongly correlated with each other.The relationship between *mpg* and log(displacement) is more linear after log transform.

(c) Split data into training set of size 20 and the rest as test set:

```
set.seed(100)
training.ind<-sample(1:dim(Auto)[1],20) #randomly choose 20 data points
data.training<-Auto[training.ind,] #training data
data.test<-Auto[-training.ind,] #test data
```

(d) Use log(*displacement*), log(*weight*), and *year* as predictor, because the others are either strongly correlated with these variables or do not seem to associate with *mpg01.*

```
library(MASS)
#fit LDA
lda.fit<-lda(mpg01~log(weight)+log(displacement)+year,data=data.training)
#predict with LDA
lda.pred<-predict(lda.fit,data.test)
```

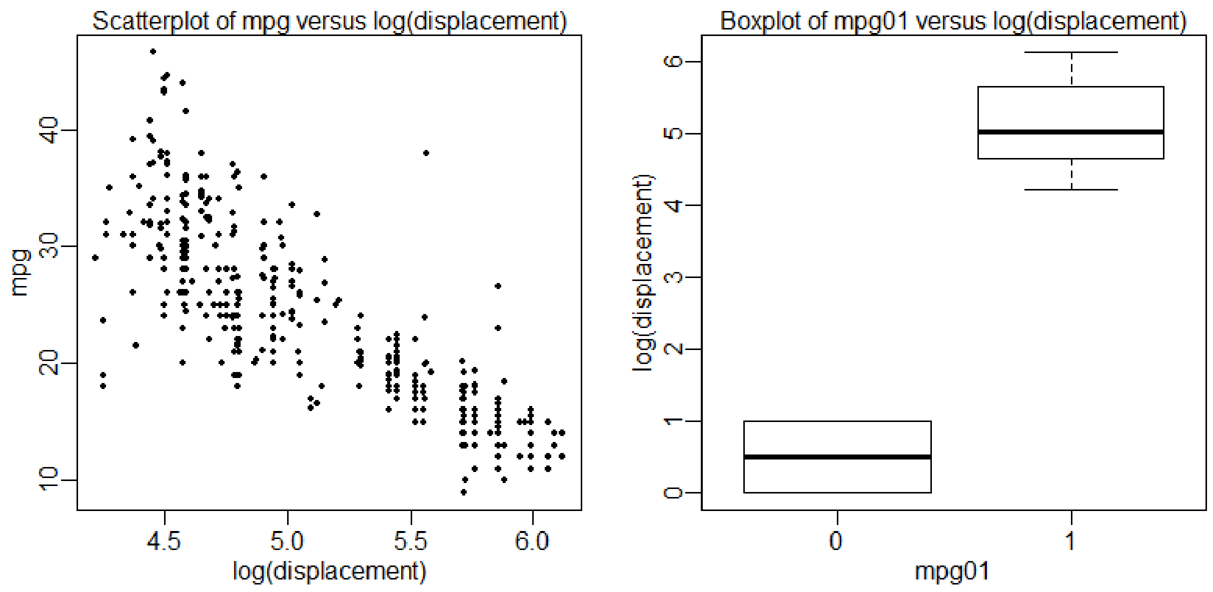

Figure 2: Left is scatterplot of *mpg* versus log(*displacement*) and right is boxplot of *mpg* versus log(*displacement*). Relationship between *mpg* and log(*displacement*) is quite linear and negative'y correlated.

```
#test with LDA
lda.class<-lda.pred$class
table.lda<-table(lda.class,data.test$mpg01) #confusion table

lda.class   0   1
        0 161   6
        1  24 181

#test error rate
1-mean(lda.class==data.test$mpg01)
[1] 0.08064516
```

Test error rate is 0.081 using LDA. We are more likely to wrongly label a car whose *mpg* is below median to the wrong class.

(e) Use the same set of predictor and QDA for classification. The code is not shown, replace "lda" to "qda" would suffice. Confusion table is

```
qda.class   0   1
        0 164  12
        1  21 175
```

and test error rate is

```
1-mean(qda.class==data.test$mpg01)
[1] 0.08870968
```

So test error rate is 0.089 using QDA.

(f) Use logistic regression to predict *mpg01* using the same set of predictors. The test error rate is 0.078.

```
#fit logistic regression
```

```
glm.fit<-glm(mpg01~log(weight)+log(displacement)+year,data=data.training,family=binomial)
#apply fitted model on test data
glm.probs<-predict(glm.fit,data.test,type="response")
glm.class<-ifelse(glm.probs>0.5,1,0) #set class by probability larger than 0.5 or not

table.glm<-table(glm.pred,data.test$mpg01) #confusion table
glm.class   0   1
        0 164   8
        1  21 179

#test error rate
1-mean(glm.class==data.test$mpg01)
[1] 0.07795699
```

(g) Before using k-nearest neighbors, we standardize predictors. Test error rate with k=1 is 0.129. After experimenting with a few values of k, k=10 gives smallest test error rate, which is 0.089.

```
#scale data
standardized.auto<-scale(Auto[,c(3,5,7)])
#k nearest neighbors
knn.pred<-knn(standardized.auto[training.ind,],standardized.auto[-training.ind,],
              Auto$mpg01[training.ind],k=1)
#test error rate
mean(knn.pred!=Auto$mpg01[-training.ind])
[1] 0.1290323
```