Part B: Choose one of Questions 10 or 11

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?

The QDA has more flexibility, so it will match the training set more closely than the LDA. The LDA will perform better on the test set because the real relationship is linear, so the QDA would have additional bias.

(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?

QDA will still perform better on the training set, but now should also perform better than LDA on the test set, since QDA will account for the additional degree of freedom in the real relationship.

(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?

Definitely increase. The LDA has an advantage when the training set is small because it is less sensitive to the fluctuations of those few data. As the size of the training set grows, the QDA is able to optimize its coefficients well, and assuming the real relationship is not linear, the QDA should eventually out-perform the LDA.

(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. The QDA will likely be biased because it will fit to training data that don't completely represent the relationship that will be observed in test data.

8. Suppose that we take a data set, divide it into equally-sized training and test sets, and then try out two different classification procedures. First we use logistic regression and get an error rate of 20 % on the training data and 30 % on the test data. Next we use 1-nearest neighbors (i.e. K = 1) and get an average error rate (averaged over both test and training data sets) of 18 %. Based on these results, which method should we prefer to use for classification of new observations? Why?

Definitely 1-nearest neighbor. The logistic regression performed more poorly on the training data, to which it has been optimized as much as possible, and yet the nearest neighbor model performs better over the entire dataset. Considering the logistic regression performed even worse on the test data, the average error rate of the logistic regression over the training and test data is 25%. This

suggests that the relationship may not even be linear, and the nearest-neighbor is a very solid method for modeling non-linear classifications, so if the real relationship is not linear, it easily explains why the nearest-neighbor method would do so much better. Everything here seems to point at using the nearest-neighbor.

9. This problem has to do with odds.

(a) On average, what fraction of people with an odds of 0.37 of defaulting on their credit card payment will in fact default?

.27

(b) Suppose that an individual has a 16 % chance of defaulting on her credit card payment. What are the odds that she will default?

.19

11. In this problem, you will develop a model to predict whether a given car gets high or low gas mileage based on the Auto data set.

(a) Create a binary variable, mpg01 , that contains a 1 if mpg contains a value above its median, and a 0 if mpg contains a value below its median. You can compute the median using the median() function. Note you may find it helpful to use the data.frame() function to create a single data set containing both mpg01 and the other Auto variables.

```
> auto$mpg01=rep(0,397)
> auto$mpg01[auto$mpg>median(auto$mpg)]=1
```

```
> auto$mpg01
  [1] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 1 1 1 1 1 0 0 0 0 0 1 1 1 1
0 0 0 0
 [38] 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0
 [75] 0 0 0 0 0 1 0 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
0 0 0 0
[112] 0 0 0 1 0 0 1 1 0 0 0 1 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 1 1 1
1 1 1 1
[149] 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 1 1 0 1 0 0 0 0 1
1 1 1 1
[186] 1 1 0 0 0 0 0 0 1 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0 1 1
1 1 1 0
[223] 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 0 0 0 1 1 1 1 1 0 0 0 0 0 0
1 0 0 0
[260] 0 0 0 0 0 0 0 1 1 1 1 0 1 1 1 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0
0 1 1 1
[297] 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1
[334] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 0
0 1 1 1
[371] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1
```

—

(b) Explore the data graphically in order to investigate the associ- ation between mpg01 and the other features. Which of the other features seem most likely to be useful in predicting mpg01 ? Scat- terplots and boxplots may be useful tools to answer this ques- tion. Describe your findings.

Horsepower clearly has the best relationship from the scatter plots. All of the mpgs over the median are on one side of the plot. Weight and acceleration are alright, but there is significant overlap between middle values. Displacement is on the cusp and the other variables don't have a terribly useful relationship with this median.

The boxplots indicate that acceleration really isn't a great predictor of mpg01, but displacement is. It also confirms horsepower and weight as good predictors, and cylinders also seems to be very strong, even though I didn't take that from the scatter plots.

I will use mpg01 ~ horsepower + weight + cylinders + displacement

---

—

(c) Split the data into a training set and a test set.

Seems like a 50/50 random sampling is appropriate enough.

```
> training_indices = sample(nrow(auto),397/2)
> train_bools = rep(F,length(auto$mpg))
> train_bools[training_indices]=T
> head(train_bools)
[1]  TRUE  TRUE  TRUE FALSE  TRUE FALSE
> length(train_bools)
[1] 397
> train_data = auto[train_bools,]
> test_data = auto[!train_bools,]
```

Actually, I changed this now, because a solution I found online suggested a different test split and I was having trouble with the KNN model, so I followed their style. I used:

```
> train <- (auto$year %% 2 == 0)
```

and then the rest the same

—

(d) Perform LDA on the training data in order to predict mpg01 using the variables that seemed most associated with mpg01 in (b). What is the test error of the model obtained?

```
> lda.fit
Call:
lda(mpg01 ~ horsepower + weight + cylinders + displacement,
data = train_data)

Prior probabilities of groups:
        0         1
0.4666667 0.5333333
```

```
        Group means:
          horsepower   weight cylinders displacement
        0  131.96939 3579.827  6.755102     268.4082
        1   77.96429 2313.598  4.071429     111.7188

        Coefficients of linear discriminants:
                              LD1
        horsepower    0.0060634365
        weight       -0.0011442212
        cylinders    -0.6390942259
        displacement  0.0004517291
```

***Test Data Error Rate:

```
        > mean(lda.pred$class!=test_data$mpg01,na.rm=T)
        [1] 0.1428571
```

(e) Perform QDA on the training data in order to predict mpg01 using the variables that seemed most associated with mpg01 in (b). What is the test error of the model obtained?

```
        > qda.fit
        Call:
        qda(mpg01 ~ horsepower + weight + cylinders + displacement,
data = train_data)

        Prior probabilities of groups:
                0         1
        0.4666667 0.5333333

        Group means:
          horsepower   weight cylinders displacement
        0  131.96939 3579.827  6.755102     268.4082
        1   77.96429 2313.598  4.071429     111.7188

        > qda.pred=predict(qda.fit,test_data,na.rm=T)
```

***Test Data Error Rate:

```
        > mean(qda.pred$class!=test_data$mpg01,na.rm=T)
        [1] 0.1483516
```

(f) Perform logistic regression on the training data in order to pre- dict mpg01 using the variables that seemed most associated with mpg01 in (b). What is the test error of the model obtained?

```
        > glm.fit=glm(mpg01 ~ horsepower + weight + cylinders +
displacement,data=train_data,family=binomial)
        > glm.probs=predict(glm.fit,test_data,type="response")
        > glm.pred=rep(0,nrow(test_data))
        > glm.pred[glm.probs>.5]=1
```

***Test Data Error Rate:

```
> mean(glm.pred!=test_data$mpg01)
[1] 0.1373626
```

(g) Perform KNN on the training data, with several values of K, in order to predict mpg01 . Use only the variables that seemed most associated with mpg01 in (b). What test errors do you obtain? Which value of K seems to perform the best on this data set?

The knn method can't handle the NA values, so

```
> set.seed(1)
> auto <- na.omit(auto)
> train_bools <- (auto$year %% 2 == 0)
> train_data = auto[train_bools,]
> test_data = auto[!train_bools,]

> train.X = cbind(auto$horsepower,auto$displacement,auto
$weight,auto$cylinders)[train_bools,]
> test.X = cbind(auto$horsepower,auto$displacement,auto
$weight,auto$cylinders)[!train_bools,]
> train.mpg01 = auto$mpg01[train_bools]
```

***Test Data Error Rates:

k = 1

```
> knn.pred = knn(train.X,test.X,train.mpg01,k=1)
> mean(knn.pred != test_data$mpg01)
[1] 0.1483516
```

k = 2

```
> knn.pred = knn(train.X,test.X,train.mpg01,k=2)
> mean(knn.pred != test_data$mpg01)
[1] 0.1593407
```

k = 3

```
> knn.pred = knn(train.X,test.X,train.mpg01,k=3)
> mean(knn.pred != test_data$mpg01)
[1] 0.1648352
```

k = 4

```
> knn.pred = knn(train.X,test.X,train.mpg01,k=4)
> mean(knn.pred != test_data$mpg0)
[1] 0.1923077
```

k = 1 looks like the best, since the error rate increases with k.