DS-6030 Homework Module 3

Matt Scheffel

DS 6030 | Spring 2022 | University of Virginia

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?

If the Bayes decision boundary is linear, we expect QDA to perform better on the training set. QDA has higher flexibility and more parameters, allowing it to fit the training set data better.

We would expect LDA to perform better on the test set, as using QDA on the test set may result in overfitting the line.

(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?

If the Bayes decision boundary is non-linear, we expect QDA to perform better on both the training set and the test set. This is due to the higher flexibility of QDA.

(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?

Typically, as the sample size n increases, we would expect the test prediction accuracy of QDA relative to LDA to improve. This is because QDA is a more flexible model than LDA.

However, in this situation, the test prediction accuracy of QDA relative to LDA tends to be uncertain. The improvement in accuracy depends on the Bayes decision boundary. If the boundary is linear, LDA typically performs better as n increases. But if the boundary is non-linear, we would expect the prediction accuracy of QDA relative to LDA to improve.

Ultimately, as the sample size n grows increasingly large, the difference in performance between QDA and LDA may become negligible.

(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. If the Bayes decision boundary for a given problem is linear, we would expect LDA to achieve a superior test error rate compared to QDA. In this situation, using LDA would simplify the model and reduce the risk of overfitting when the sample size is small or when there are many predictors relative to the sample size. QDA is more flexible and may result in overfitting when the sample size is small or the number of predictors is large. Ultimately, when the Bayes decision boundary is linear, LDA would likely result in a superior test error rate in comparison to QDA.

13. This question should be answered using the `Weekly` data set, which is part of the ISLR2 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?

```
#numerical summaries
library(ISLR2)
library(MASS)
library(class)

summary(Weekly)
```

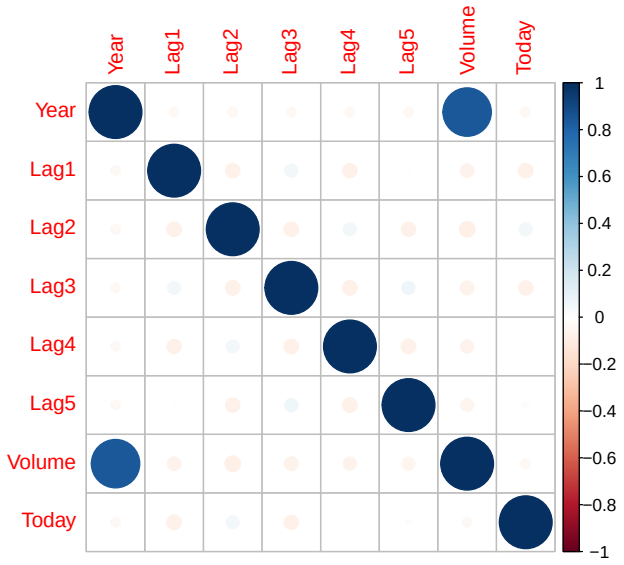
```
#>       Year           Lag1               Lag2               Lag3
#>  Min.   :1990   Min.   :-18.1950   Min.   :-18.1950   Min.   :-18.1950
#>  1st Qu.:1995   1st Qu.: -1.1540   1st Qu.: -1.1540   1st Qu.: -1.1580
#>  Median :2000   Median :  0.2410   Median :  0.2410   Median :  0.2410
#>  Mean   :2000   Mean   :  0.1506   Mean   :  0.1511   Mean   :  0.1472
#>  3rd Qu.:2005   3rd Qu.:  1.4050   3rd Qu.:  1.4090   3rd Qu.:  1.4090
#>  Max.   :2010   Max.   : 12.0260   Max.   : 12.0260   Max.   : 12.0260
#>       Lag4               Lag5              Volume            Today
#>  Min.   :-18.1950   Min.   :-18.1950   Min.   :0.08747   Min.   :-18.1950
#>  1st Qu.: -1.1580   1st Qu.: -1.1660   1st Qu.:0.33202   1st Qu.: -1.1540
#>  Median :  0.2380   Median :  0.2340   Median :1.00268   Median :  0.2410
#>  Mean   :  0.1458   Mean   :  0.1399   Mean   :1.57462   Mean   :  0.1499
#>  3rd Qu.:  1.4090   3rd Qu.:  1.4050   3rd Qu.:2.05373   3rd Qu.:  1.4050
#>  Max.   : 12.0260   Max.   : 12.0260   Max.   :9.32821   Max.   : 12.0260
#>  Direction
#>  Down:484
#>  Up  :605
#>
#>
#>
#>
```

```
cor(Weekly[ ,-9])
```

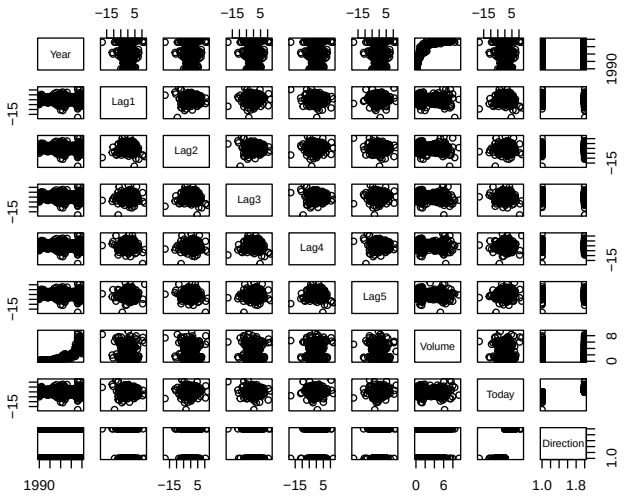
```
#>               Year         Lag1        Lag2        Lag3         Lag4
#> Year    1.00000000 -0.032289274 -0.03339001 -0.03000649 -0.031127923
#> Lag1   -0.03228927  1.000000000 -0.07485305  0.05863568 -0.071273876
#> Lag2   -0.03339001 -0.074853051  1.00000000 -0.07572091  0.058381535
#> Lag3 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

```
#> Volume -0.058517414  1.00000000 -0.033077783
#> Today   0.011012698 -0.03307778  1.000000000
```

```
#graphical summaries
library(corrplot)

corrplot(cor(Weekly[,-9]))
```




```
pairs(Weekly)
```



Most of the variables tend to have no correlation or pattern aside from a noticeable relationship between Volume and Year.

(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?

```
weekly_log_reg <- glm(Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume,
                      data = Weekly,
                      family = binomial)

summary(weekly_log_reg)
```

```
#>
#> Call:
#> glm(formula = Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 +
#>     Volume, 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
```

The only predictor that appears to be statistically significant (at the 0.05 level) is Lag2.

(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.

```
weekly_log_prob = predict(weekly_log_reg, type = "response")
weekly_log_pred = rep("Down", length(weekly_log_prob))
weekly_log_pred[weekly_log_prob > 0.5] <- "Up"

table(weekly_log_pred, Weekly$Direction)
```

```
#>
#> weekly_log_pred Down  Up
#>            Down   54  48
#>            Up    430 557
```

```
correct_preds = mean(weekly_log_pred == Weekly$Direction)
correct_preds
```

```
#> [1] 0.5610652
```

The confusion matrix is telling us:

Approximately 56.11% of the responses are predicted correctly. The model correctly predicted 54 out of 484 down days accurately (approx. 11%). The model correctly predicted 557 out of 605 up days (approx. 92%).

(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).

```
training_data = (Weekly$Year < 2009)

weekly_heldout <- Weekly[!training_data,]

model_1_fit <- glm(Direction ~ Lag2, data = Weekly,family = binomial, subset = training_data)

weekly_log_prob= predict(model_1_fit, weekly_heldout, type = "response")
weekly_log_pred = rep("Down", length(weekly_log_prob))
weekly_log_pred[weekly_log_prob > 0.5] = "Up"

direction_heldout = Weekly$Direction[!training_data]

table(weekly_log_pred, direction_heldout)
```

```
#>                direction_heldout
#> weekly_log_pred Down Up
#>            Down    9  5
#>            Up     34 56
```

```
mean(weekly_log_pred == direction_heldout)
```

```
#> [1] 0.625
```

(e) Repeat (d) using LDA.

```
model_lda_fit <-lda(Direction ~ Lag2, data = Weekly, subset = training_data)
model_lda_pred <-predict(model_lda_fit, weekly_heldout)

table(model_lda_pred$class, direction_heldout)
```

```
#>       direction_heldout
#>        Down Up
#>   Down    9  5
#>   Up     34 56
```

```
mean(model_lda_pred$class==direction_heldout)
```

```
#> [1] 0.625
```

(f) Repeat (d) using QDA.

```
model_qda_fit <-qda(Direction ~ Lag2, data = Weekly, subset = training_data)
model_qda_pred <-predict(model_qda_fit, weekly_heldout)

table(model_qda_pred$class, direction_heldout)
```

```
#>       direction_heldout
#>        Down Up
#>   Down    0  0
#>   Up     43 61
```

```
mean(model_qda_pred$class==direction_heldout)
```

```
#> [1] 0.5865385
```

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

```
knn_train <- as.matrix(Weekly$Lag2[training_data])
knn_test <- as.matrix(Weekly$Lag2[!training_data])
```

```
direction_train = Weekly$Direction[training_data]

set.seed(1)
knn_pred <- knn(knn_train, knn_test, direction_train, k = 1)

table(knn_pred, direction_heldout)
```

```
#>         direction_heldout
#> knn_pred Down Up
#>     Down   21 30
#>     Up     22 31
```

```
mean(knn_pred == direction_heldout)
```

```
#> [1] 0.5
```

(h) Repeat (d) using naive Bayes. (skip this exercise)

#skip

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

The Logistic Regression and Linear Discriminant Analysis methods appear to provide the best results on this data as both produce accuracy rates of 62.5%.

(j) 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.

```
new_model_1 <- glm(Direction ~ Lag2:Lag1, data = Weekly, family = binomial, subset = training_data)

weekly_log_prob= predict(new_model_1, weekly_heldout, type = "response")
weekly_log_pred = rep("Down", length(weekly_log_prob))
weekly_log_pred[weekly_log_prob > 0.5] = "Up"

direction_heldout = Weekly$Direction[!training_data]

table(weekly_log_pred, direction_heldout)
```

```
#>                direction_heldout
#> weekly_log_pred Down Up
#>            Down    1  1
#>            Up     42 60
```

```
mean(weekly_log_pred == direction_heldout)
```

```
#> [1] 0.5865385
```

```
model_lda_fit2 <-lda(Direction ~ Lag2:Lag1, data = Weekly, subset = training_data)
model_lda_pred2 <-predict(model_lda_fit2, weekly_heldout)

table(model_lda_pred2$class, direction_heldout)
```

```
#>       direction_heldout
#>        Down Up
#>   Down    0  1
#>   Up     43 60
```

```
mean(model_lda_pred2$class==direction_heldout)
```

```
#> [1] 0.5769231
```

```
model_qda_fit2 <-qda(Direction ~ Lag2 + sqrt(abs(Lag2)), data = Weekly, subset = training_data)
model_qda_pred2 <-predict(model_qda_fit2, weekly_heldout)

table(model_qda_pred2$class, direction_heldout)
```

```
#>       direction_heldout
#>        Down Up
#>   Down   12 13
#>   Up     31 48
```

```
mean(model_qda_pred2$class==direction_heldout)
```

```
#> [1] 0.5769231
```

```
knn_train2 <- as.matrix(Weekly$Lag2[training_data])
knn_test2 <- as.matrix(Weekly$Lag2[!training_data])
direction_train = Weekly$Direction[training_data]

set.seed(1)
knn_pred2 <- knn(knn_train2, knn_test2, direction_train, k = 5)

table(knn_pred2, direction_heldout)
```

```
#>          direction_heldout
#> knn_pred2 Down Up
#>      Down   16 21
#>      Up     27 40
```

```
mean(knn_pred2 == direction_heldout)
```

```
#> [1] 0.5384615
```

```
knn_train3 <- as.matrix(Weekly$Lag2[training_data])
knn_test3 <- as.matrix(Weekly$Lag2[!training_data])
direction_train = Weekly$Direction[training_data]

set.seed(1)
knn_pred3 <- knn(knn_train3, knn_test3, direction_train, k = 10)

table(knn_pred3, direction_heldout)
```

```
#>          direction_heldout
#> knn_pred3 Down Up
#>      Down   17 21
#>      Up     26 40
```

```
mean(knn_pred3 == direction_heldout)
```

```
#> [1] 0.5480769
```

After experimenting with different combinations of predictors, including possible transformations and interactions, for each of the methods, it appears that the original models produce the best results and highest accuracy rates.

14. 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.

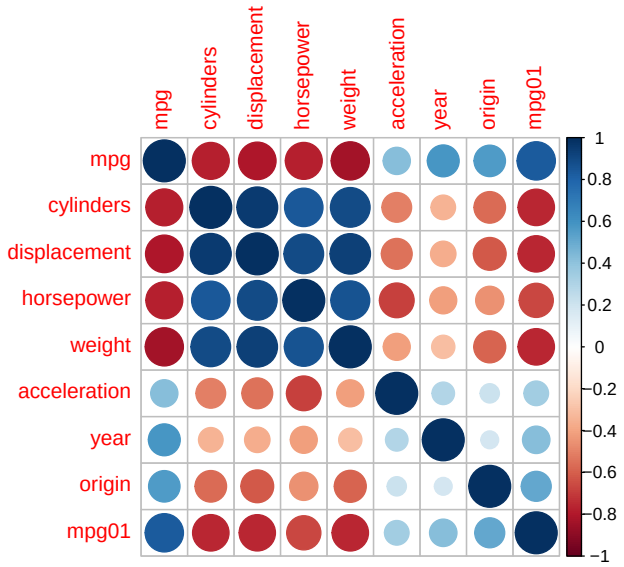
```
attach(Auto)
summary(Auto)
```

```
#>       mpg          cylinders      displacement     horsepower        weight
#>  Min.   : 9.00   Min.   :3.000   Min.   : 68.0   Min.   : 46.0   Min.   :1613
#>  1st Qu.:17.00   1st Qu.:4.000   1st Qu.:105.0   1st Qu.: 75.0   1st Qu.:2225
#>  Median :22.75   Median :4.000   Median :151.0   Median : 93.5   Median :2804
#>  Mean   :23.45   Mean   :5.472   Mean   :194.4   Mean   :104.5   Mean   :2978
#>  3rd Qu.:29.00   3rd Qu.:8.000   3rd Qu.:275.8   3rd Qu.:126.0   3rd Qu.:3615
#>  Max.   :46.60   Max.   :8.000   Max.   :455.0   Max.   :230.0   Max.   :5140
#>
#>   acceleration        year           origin                      name
#>  Min.   : 8.00   Min.   :70.00   Min.   :1.000   amc matador       :  5
#>  1st Qu.:13.78   1st Qu.:73.00   1st Qu.:1.000   ford pinto        :  5
#>  Median :15.50   Median :76.00   Median :1.000   toyota corolla    :  5
#>  Mean   :15.54   Mean   :75.98   Mean   :1.577   amc gremlin       :  4
#>  3rd Qu.:17.02   3rd Qu.:79.00   3rd Qu.:2.000   amc hornet        :  4
#>  Max.   :24.80   Max.   :82.00   Max.   :3.000   chevrolet chevette:  4
#>                                                  (Other)           :365
```

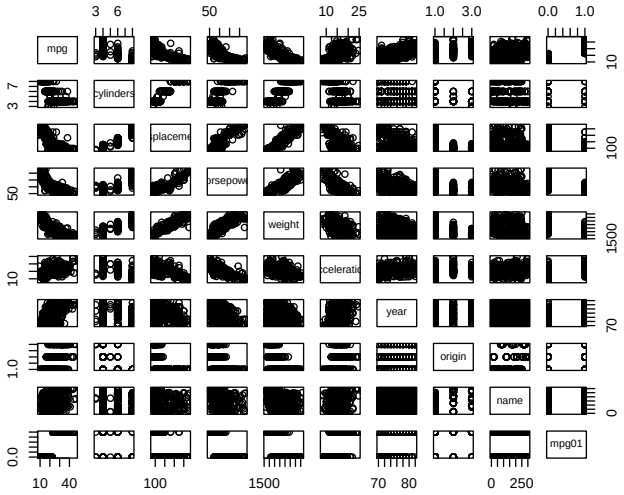
```
mpg01 <- rep(0, length(Auto$mpg))
mpg01[Auto$mpg > median(Auto$mpg)] <- 1

auto_new = data.frame(Auto, mpg01)
```

(b) Explore the data graphically in order to investigate the association between `mpg01` and the other features. Which of the other features seem most likely to be useful in predicting `mpg01`? Scatterplots and boxplots may be useful tools to answer this question. Describe your findings.

```
corrplot(cor(auto_new[,-9]))
```




```
pairs(auto_new)
```



From the visualizations, it appears that mpg01 has strong negative correlation with cylinders, displacement, horsepower, and weight. There is slight positive correlation with acceleration, year, and origin. (Obviously mpg also has strong correlation with mpg01.)

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

```
training_data_2 <- (auto_new$year %% 2 == 0)
auto_train <- auto_new[training_data_2,]
auto_test <- auto_new[-training_data_2,]

auto_train2 <- auto_train[complete.cases(auto_train),]
auto_test2 <- auto_test[complete.cases(auto_test),]
```

(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?

```
auto_lda_fit <- lda(mpg01~displacement+horsepower+weight+year+cylinders+origin, data=auto_train)
auto_lda_pred <- predict(auto_lda_fit, auto_test)

mean(auto_lda_pred$class != auto_test$mpg01)
```

```
#> [1] 0.08439898
```

```
testerror = 1 - mean(auto_lda_pred$class != auto_test$mpg01)
testerror
```

```
#> [1] 0.915601
```

```
table(auto_lda_pred$class, auto_test$mpg01)
```

```
#>
#>       0   1
#>   0 169   7
#>   1  26 189
```

The test error rate is 8.4%.

(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?

```
auto_qda_fit <- qda(mpg01~displacement+horsepower+weight+year+cylinders+origin, data=auto_train)
auto_qda_pred <- predict(auto_qda_fit, auto_test)

mean(auto_qda_pred$class != auto_test$mpg01)
```

```
#> [1] 0.09974425
```

```
testerror = 1 - mean(auto_qda_pred$class != auto_test$mpg01)
testerror
```

```
#> [1] 0.9002558
```

```
table(auto_qda_pred$class, auto_test$mpg01)
```

```
#>
#>       0   1
#>   0 176  20
#>   1  19 176
```

The test error rate is 9.9%.

(f) Perform logistic regression 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?

```
auto_log_reg <- glm(mpg01 ~ displacement + horsepower + weight + year + cylinders + origin, data = auto
auto_log_prob = predict(auto_log_reg, auto_test, type = "response")
auto_log_pred = rep(0, length(auto_log_prob))
auto_log_pred[auto_log_prob > 0.5] = 1

mean(auto_log_pred != auto_test$mpg01)
```

```
#> [1] 0.08439898
```

```
testerror = 1 - mean(auto_log_pred != auto_test$mpg01)
testerror
```

```
#> [1] 0.915601
```

```
table(auto_log_pred, auto_test$mpg01)
```

```
#>
#> auto_log_pred   0   1
#>             0 174  12
```

```
#>             1  21 184
```

The test error is 8.4%.

(g) Perform naive Bayes 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? (skip this exercise)

#skip

(h) 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?

```
auto_knn_train = cbind(displacement, horsepower, weight, cylinders, year, origin)[training_data_2,]
auto_knn_test=cbind(displacement,horsepower,weight,cylinders, year, origin)[-training_data_2,]
set.seed(1)
autok.pred=knn(auto_knn_train, auto_knn_test, auto_train$mpg01, k = 1)
mean(autok.pred != auto_test$mpg01)
```

```
#> [1] 0.07161125
```

```
auto_knn_train = cbind(displacement, horsepower, weight, cylinders, year, origin)[training_data_2,]
auto_knn_test=cbind(displacement,horsepower,weight,cylinders, year, origin)[-training_data_2,]
set.seed(1)
autok.pred=knn(auto_knn_train, auto_knn_test, auto_train$mpg01, k = 5)
mean(autok.pred != auto_test$mpg01)
```

```
#> [1] 0.112532
```

```
auto_knn_train = cbind(displacement, horsepower, weight, cylinders, year, origin)[training_data_2,]
auto_knn_test=cbind(displacement,horsepower,weight,cylinders, year, origin)[-training_data_2,]
set.seed(1)
autok.pred=knn(auto_knn_train, auto_knn_test, auto_train$mpg01, k = 10)
mean(autok.pred != auto_test$mpg01)
```

```
#> [1] 0.1227621
```

K = 1 has the lowest error rate of 7.16%. For this model it appears that as K increases, the error rate also increases.