Classifiers: LDA, QDA, KNN Model Evaluation

Chris Schmidt

10/24/2019

Evaluating AutoData using Linear Discriminant Analysis, Quadratic Discriminant Analysis, Logistic Regression and KNN

This project comes from *An Introduction to Statistical Learning with Applications in R*, by James, Witten, Hastie, and Tibshirani, using the problem set in chapter 4, problem 11.

There is a discussion and examination of the mathematics and intuition of the LDA, QDA, Logistic Regression, and Generalized Linear Model processes and models are built for each of these processes using the Auto data set from the ISLR package in R to provide comparison information between them in terms of accuracy given the same problem.

In this problem we 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.

1. Take a look at the median for mpg. Use the *median()* function on the argument *Auto$mpg* to generate the median value for miles per gallon of the vehicles in the Auto data set.

```
median(Auto$mpg)
```

```
## [1] 22.75
```

```
mpg01 <- (Auto$mpg > median(Auto$mpg))*1
```

2. Create variable mpg01 as described above

```
summary(mpg01)
```

3. Take a look at the variable *mpg01* using the functions summary(), str(), and head() to ensure output makes sense for mpg01 variable..

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     0.0     0.0     0.5     0.5     1.0     1.0
```

```
str(mpg01)
```

```
##  num [1:392] 0 0 0 0 0 0 0 0 0 0 ...
```

```
head(Auto$mpg, 30)
```

```
##  [1] 18 15 18 16 17 15 14 14 14 15 15 14 15 14 24 22 18 21 27 26 25 24 25 26 21
## [26] 10 10 11  9 27
```

```
head(mpg01, 30)
```

```
##  [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
```

With the median value of mpg at 22.75 we can see that the output above makes sense.

4.Create a data frame AutoData using the new variable *mpg01* from the Auto data set then look at the dimension and summary of the data frame. We have 392 cases and 10 attributes consisting of nine columns of numeric data and one column of factor data for the vehicle name.

The 'cylinders' data will need be encoded as a factor variable since it is categorical data.

```
AutoData <- data.frame(mpg01, Auto)
dim(AutoData)
```

```
## [1] 392  10
```

```
summary(AutoData)
```

```
##      mpg01          mpg          cylinders      displacement     horsepower
##  Min.   :0.0   Min.   : 9.00   Min.   :3.000   Min.   : 68.0   Min.   : 46.0
##  1st Qu.:0.0   1st Qu.:17.00   1st Qu.:4.000   1st Qu.:105.0   1st Qu.: 75.0
##  Median :0.5   Median :22.75   Median :4.000   Median :151.0   Median : 93.5
##  Mean   :0.5   Mean   :23.45   Mean   :5.472   Mean   :194.4   Mean   :104.5
##  3rd Qu.:1.0   3rd Qu.:29.00   3rd Qu.:8.000   3rd Qu.:275.8   3rd Qu.:126.0
##  Max.   :1.0   Max.   :46.60   Max.   :8.000   Max.   :455.0   Max.   :230.0
##
##      weight      acceleration        year           origin
##  Min.   :1613   Min.   : 8.00   Min.   :70.00   Min.   :1.000
##  1st Qu.:2225   1st Qu.:13.78   1st Qu.:73.00   1st Qu.:1.000
##  Median :2804   Median :15.50   Median :76.00   Median :1.000
##  Mean   :2978   Mean   :15.54   Mean   :75.98   Mean   :1.577
##  3rd Qu.:3615   3rd Qu.:17.02   3rd Qu.:79.00   3rd Qu.:2.000
##  Max.   :5140   Max.   :24.80   Max.   :82.00   Max.   :3.000
##
##                  name
##  amc matador       :  5
##  ford pinto        :  5
##  toyota corolla    :  5
##  amc gremlin       :  4
##  amc hornet        :  4
##  chevrolet chevette:  4
##  (Other)           :365
```

```
str(AutoData)
```

```
## 'data.frame':    392 obs. of  10 variables:
##  $ mpg01       : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ mpg         : num  18 15 18 16 17 15 14 14 14 15 ...
##  $ cylinders   : num  8 8 8 8 8 8 8 8 8 8 ...
##  $ displacement: num  307 350 318 304 302 429 454 440 455 390 ...
##  $ horsepower  : num  130 165 150 150 140 198 220 215 225 190 ...
##  $ weight      : num  3504 3693 3436 3433 3449 ...
##  $ acceleration: num  12 11.5 11 12 10.5 10 9 8.5 10 8.5 ...
##  $ year        : num  70 70 70 70 70 70 70 70 70 70 ...
```

```
##  $ origin      : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ name        : Factor w/ 304 levels "amc ambassador brougham",..: 49 36 231 14 161 141 54 223 241 1
```

(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 predictiong mpg01 ? Scatterplots and boxplots may be useful tools to answer this question. Describe your findings.

```
cor(AutoData[, -10])
```

1. Look at the correlations between variables.

```
##                    mpg01        mpg  cylinders displacement horsepower
## mpg01          1.0000000  0.8369392 -0.7591939   -0.7534766 -0.6670526
## mpg            0.8369392  1.0000000 -0.7776175   -0.8051269 -0.7784268
## cylinders     -0.7591939 -0.7776175  1.0000000    0.9508233  0.8429834
## displacement  -0.7534766 -0.8051269  0.9508233    1.0000000  0.8972570
## horsepower    -0.6670526 -0.7784268  0.8429834    0.8972570  1.0000000
## weight        -0.7577566 -0.8322442  0.8975273    0.9329944  0.8645377
## acceleration   0.3468215  0.4233285 -0.5046834   -0.5438005 -0.6891955
## year           0.4299042  0.5805410 -0.3456474   -0.3698552 -0.4163615
## origin         0.5136984  0.5652088 -0.5689316   -0.6145351 -0.4551715
##                   weight acceleration       year     origin
## mpg01         -0.7577566    0.3468215  0.4299042  0.5136984
## mpg           -0.8322442    0.4233285  0.5805410  0.5652088
## cylinders      0.8975273   -0.5046834 -0.3456474 -0.5689316
## displacement   0.9329944   -0.5438005 -0.3698552 -0.6145351
## horsepower     0.8645377   -0.6891955 -0.4163615 -0.4551715
## weight         1.0000000   -0.4168392 -0.3091199 -0.5850054
## acceleration  -0.4168392    1.0000000  0.2903161  0.2127458
## year          -0.3091199    0.2903161  1.0000000  0.1815277
## origin        -0.5850054    0.2127458  0.1815277  1.0000000
```

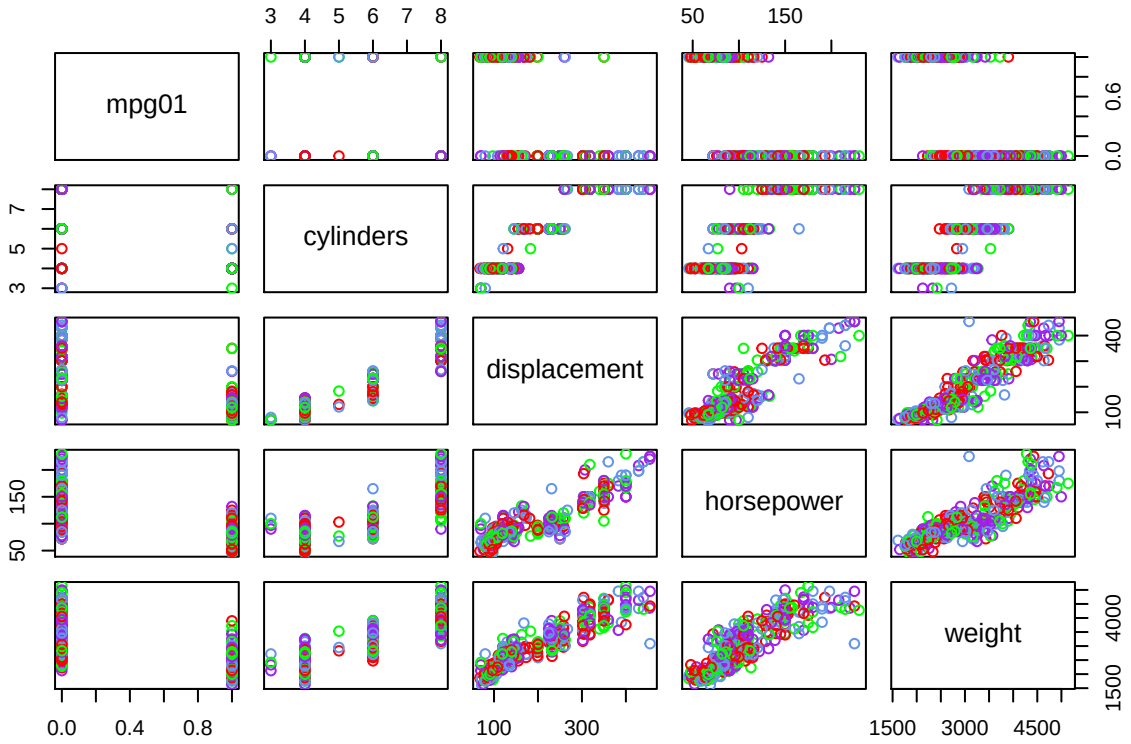
```
AutoData$name <- NULL
head(AutoData)
```

```
##   mpg01 mpg cylinders displacement horsepower weight acceleration year origin
## 1     0  18         8          307        130   3504         12.0   70      1
## 2     0  15         8          350        165   3693         11.5   70      1
## 3     0  18         8          318        150   3436         11.0   70      1
## 4     0  16         8          304        150   3433         12.0   70      1
## 5     0  17         8          302        140   3449         10.5   70      1
## 6     0  15         8          429        198   4341         10.0   70      1
```

The strongest relationships between **mpg01** and the oher variables is with **cylinders**, **displacement**, **horsepower**, and **weight**.

```
pairs(~mpg01 + cylinders + displacement + horsepower + weight, AutoData, col = c("red", "cornflowerblue
```

2. Run the pairs plot to visualize the relationships.



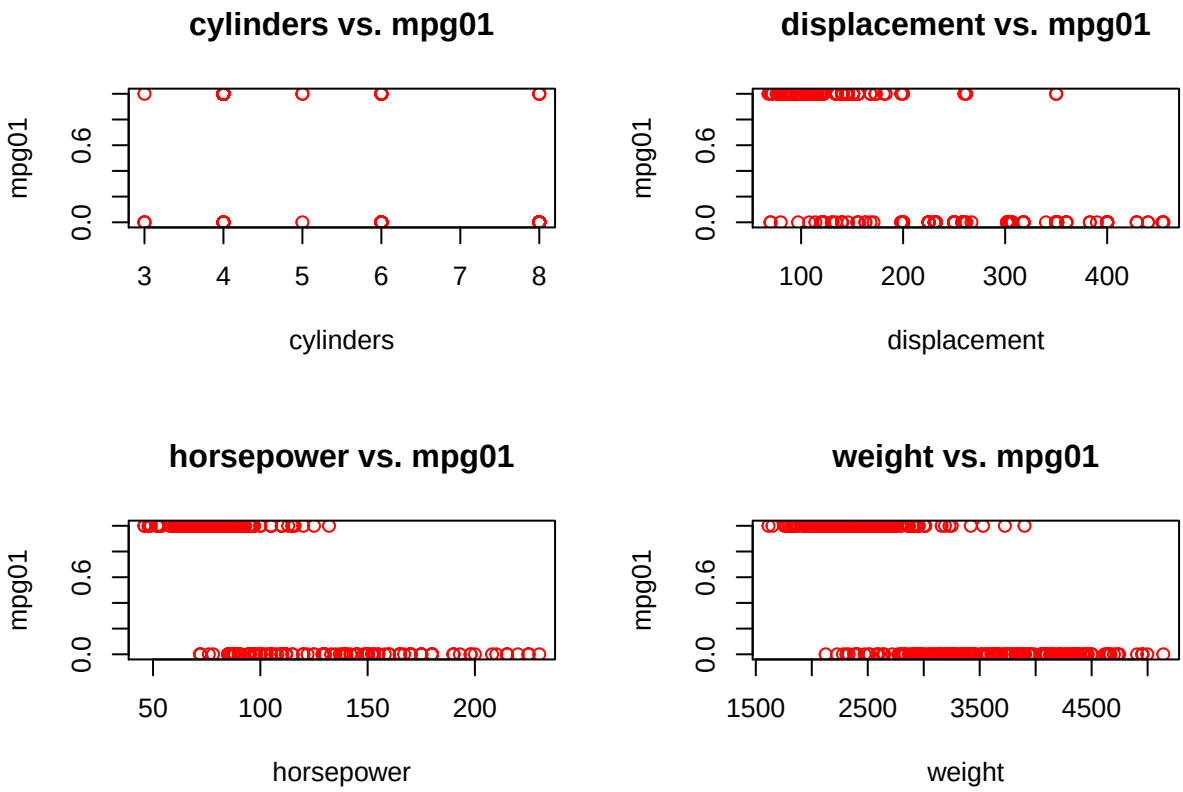

```
cylinders = as.factor(AutoData$cylinders)
summary(cylinders)
```

3.Encode the cylinders variable as a factor.

```
##   3   4   5   6   8
##   4 199   3  83 103
```

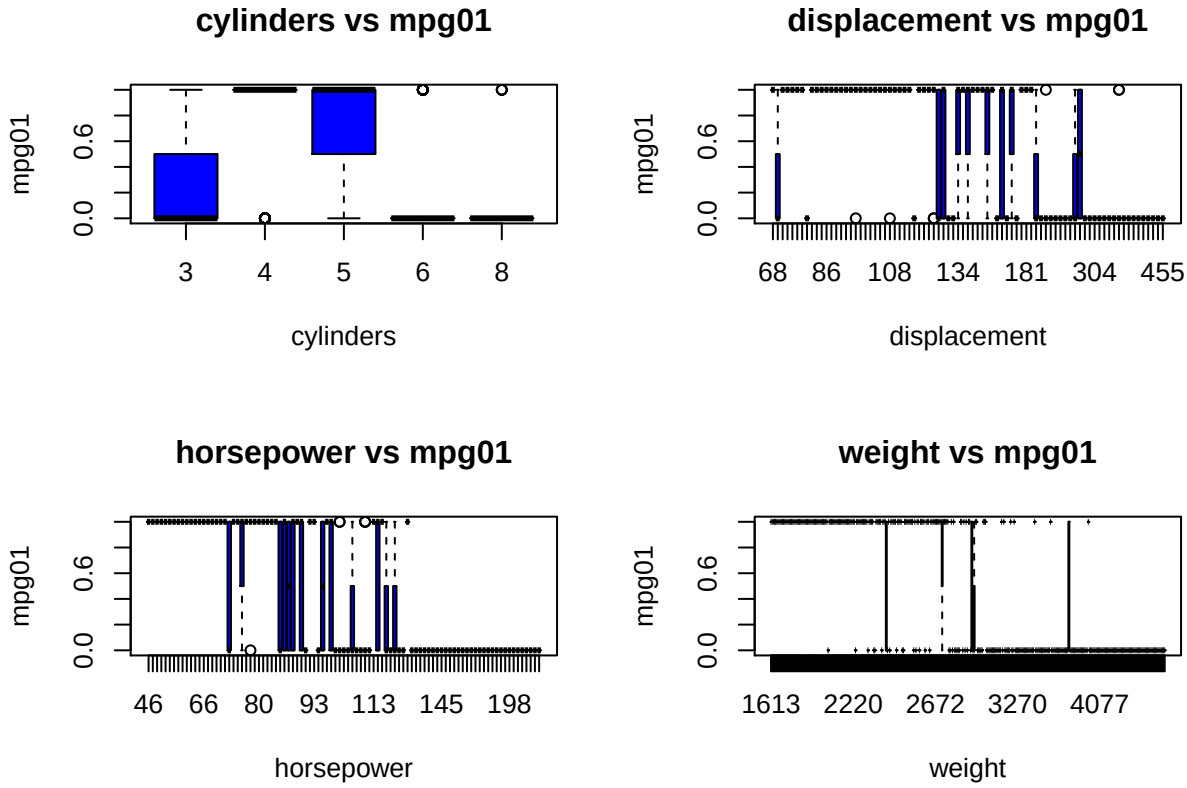
4. Show plots of the different variables versus *mpg01* We can look at pairs plots as shown below.

```
par(mfrow = c(2,2))
plot(AutoData$cylinders, mpg01, col = "red", xlab = "cylinders", main = "cylinders vs. mpg01")
plot(AutoData$displacement, mpg01, col = "red", xlab = "displacement", main = "displacement vs. mpg01")
plot(AutoData$horsepower, mpg01, col = "red", xlab = "horsepower", main = "horsepower vs. mpg01")
plot(AutoData$weight, mpg01, col = "red", xlab = "weight", main = "weight vs. mpg01")
```



And box plots of the same combinations as shown below which are not very visually useful.

```
par(mfrow = c(2,2))
boxplot(mpg01 ~ cylinders, data = AutoData, main = "cylinders vs mpg01",
        xlab = "cylinders", col='blue')
boxplot(mpg01 ~ displacement, data = AutoData, main = "displacement vs mpg01",
        xlab = "displacement", col='blue')
boxplot(mpg01 ~ horsepower, data = AutoData, main = "horsepower vs mpg01",
        xlab = "horsepower", col='blue')
boxplot(mpg01 ~ weight, data = AutoData, main = "weight vs mpg01",
        xlab = "weight", col='blue')
```



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

The data is split into a 70% training data set and the 30% balance held out as the test set using the *CreateDataPartition()* function with the response *mpg01*.

```
set.seed(999)
inTraining <- createDataPartition(AutoData$mpg01, p = 0.7, list = FALSE)

training <- AutoData[inTraining, ]
testing  <- AutoData[-inTraining, ]
```

```
dim(training)
```

1. Verify the dimensions of the training and testing sets.

```
## [1] 276   9
```

```
dim(testing)
```

```
## [1] 116   9
```

(d) Perform Linear Discriminant Analysis 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 ?

Linear discriminant analysis models the probability distribution of the predictors X separately in each of the response classes Y, i.e. we want to find $Pr(X = x|Y = k)$ and Bayes Theorem is used to flip these around

into estimates for $Pr(Y = k|X = x)$. When the distributions are Gaussian this model is very close in form to logistic regression. The reasons for using LDA over logistic regression include the facts:

- when the classes are well separated, the parameter estimates for the logistic regression model can be unstable.
- if n is small and the distribution of the predictors $X \sim N(\mu, \sigma^2)$, LDA is more stable than logistic regression.
- If we have more than two response classes Y, LDA is more popular.

Using Bayes Theorem for Classification If $k \geq 2$ and we want to classify an observation into one of K classes where the qualitative response variable Y can take on one of K distinct and unordered values. We let π_k be the *prior* probability that a randomly chosen observation comes from the kth class and let $f_k(x) \equiv Pr(X = x|Y = k)$ be the density function of X for an observation from the kth class. $f_k(x)$ is relatively large if there is a high probability that an observation in the kth class has $X \approx x$ and $f_k(x)$ is relatively small if it is very unlikely that an observation in the kth class has $X \approx x$.

Bayes Theorem states that

$$Pr(Y = k|X = x) = \frac{\pi_k f_k(x)}{\sum_{l=1}^{K} \pi_l f_l(x)}$$

Letting $p_k(x) = Pr(Y = k|X)$ we see that we can simply plug in estimates of π_k and $f_k(X)$ into the formula which can be generated with the software that then takes care of the rest. We refer to $p_k(x)$ as the *posterior* probability that an observation $X = x$ belongs to the kth class given the predictor value for that observation.

Estimating π_k is easy if we have a random sample of Y's from the population but estimating $f_k(X)$ is more difficult. However, if we have an estimate for $f_k(x)$ then we can build a classifier that approximates the Bayes classifier.

By assuming that $X = (X_1, X_2, ..., X_p)$ is drawn from a multivariate Gaussian distribution, with a class specific mean vector and a common covariance matrix which we can write as $X \sim N(\mu, \Sigma)$ to indicate that p has a multivariate Gaussian distribution. $E(X) = \mu$ is the mean of the X vector with p components and $Cov(X) = \Sigma$ is the $p \times p$ covariance matrix of X.

Formally, the multivariate Gaussian density is

$$f(x) = \frac{1}{(2\pi)^{p/2}|\Sigma|^{1/2}} exp\Big(-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)\Big)$$

plugging the density function for the kth class, $f_k(X = x)$ into

$$Pr(Y = k|X = x) = \frac{\pi_k f_k(x)}{\sum_{l=1}^{K} \pi_l f_l(x)}$$

and applying some algebra we see that the Bayes classifier assigns $X = x$ to the class for which

$$\delta_k(x) = x^T \Sigma^{-1} \mu_k - \frac{1}{2} \mu_k^T \Sigma^{-1} \mu_k + \log \pi_k$$

is the largest. The Bayes decision boundaries represent the set of values x for which $\delta_k(x) = \delta_l(x)$. In other words for which

$$x^T \Sigma^{-1} \mu_k - \frac{1}{2} \mu_k^T \Sigma^{-1} \mu_k = x^T \Sigma^{-1} \mu_l - \frac{1}{2} \mu_l^T \Sigma^{-1} \mu_l, \text{ for } k \neq l$$

The log π_k term has disappeared because each of the three classes has the same number of training observations, thus π_k is the same for each class. To estimate $\mu_1, ..., \mu_k, \pi_1, ..., \pi_k$ and Σ we use similar conventions for the case where $p = 1$

$$\hat{\mu}_k = \frac{1}{n_k} \sum_{i:y_i=k} x_i$$

$$\hat{\Sigma} = \frac{1}{n-K} \sum_{k=1}^{K} \sum_{i:y_i=k} (x_i - \hat{\mu}_k)^2$$

$$\hat{\pi}_k = \frac{n_k}{n}$$

The estimates are plugged into

$$\delta_k(x) = x^T \Sigma^{-1} \mu_k - \frac{1}{2} \mu_k^T \Sigma^{-1} \mu_k + \log \pi_k$$

in order to assign a new observation $X = x$ to the class for which $\hat{\delta}_k(x)$ is the largest. This is a linear function of x so the LDA decision rule depends on x only through a linear combination of the elements.

The output for LDA often uses a *confusion matrix* to display the True status versus the predicted status for the qualitative response variable. Elements on the diagonal represent correct predictions and off-diagonal represent misclassifications.

This introduces the terms *sensitivity* and *specificity* to characterize the performance of a classifier. Sensitivity is the percentage of correctly specified positive responses identified while specificity is the percentage of correctly specified negative responses that are identified. We call the true positive rate the *sensitivity* and the false positive rate, 1 - *specificity*

The Bayes classifier works by assigning an observation to the class for which the posterior probability $p_k(X)$ is the largest. If we have two classes, say "*wrong*" and "*right*" we assign the observation to the "*wrong*" class if

$$Pr(\text{ wrong} = \text{Yes } | X = x) > 0.5$$

This creates a threshold of 50% for the *posterior* probability of default in order to assign an observation to the "*wrong*" class. If we have concerns about mislabeling the prediction for the "*wrong*" class we can lower this threshold. We could, for example, label an observation with a posterior probability of being in the "*wrong*" class about 20% to the "*wrong*" class

$$Pr(\text{ wrong} = \text{Yes } | X = x) > 0.2$$

We use the *receiver operating characteristics* curve, ROC curve, to simultaneously display the two types of errors for all possible thresholds where the overall performance of the classifier is given by the area under the ROC curve (the AUC) where the larger the percentage, the better the classifier.

To build out Linear Discriminant Analysis model we use the *lda()* function with *mpg01* as our response variable on the predictors *cylinders, displacement, horsepower* and *weight* using the training data and name this model *lda_m1*.

We also output the confusion matrix using the *table* function on the arguments *lda.pred* and the testing data with *mpg01* as the response.

```
lda_m1 <- lda(mpg01 ~ cylinders + displacement + horsepower
              + weight, data = training)

lda.pred <- predict(lda_m1, testing)$class

table(lda.pred, testing$mpg01)
```

```
##
## lda.pred  0  1
##        0 51  2
##        1  7 56
```

1. Compute error for lda model m1. We want to know the error for the model to understand its accuracy.

```
error_lda <- mean(lda.pred != testing$mpg01)
error_lda
```

```
## [1] 0.07758621
```

(e) Perform Quadratic Discriminant Analysis 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 assumes the observations come from a Gaussian distribution like LDA but QDA assumes each class has its own covariance matrix. QDA assumes that an observation from the kth class is of the form $X \sim N(\mu_k, \Sigma_k)$, where Σ_k is a covariance matrix for the kth class.

In this assumption, the Bayes classifier assigns an observation $X = x$ to the class for which

$$\begin{aligned}\delta_k(x) &= -\frac{1}{2}(x-\mu_k)^T\Sigma_k^{-1}(x-\mu_k) - \frac{1}{2}\log|\Sigma_k| + \log\pi_k \\ &= -\frac{1}{2}x^T\Sigma_k^{-1}x + x^T\Sigma_k^{-1}\mu_k - \frac{1}{2}\mu_k^T\Sigma_k^{-1}\mu_k - \log|\Sigma_k| + \log\pi_k\end{aligned}$$

is the largest. The QDA classifier plugs estimates for Σ_k, μ_k, and π_k into the equation above and then assigning $X = x$ to the class for which the quantity is largest. Since x appears as a quadratic equation, we call this classifier QDA.

The reasons for choosing LDA over QDA or vice versa have to do with the bias-variance tradeoff. When there are p predictors, estimating the covariance matrix requires estimating $p(p+1)/2$ parameters. QDA estimates a separate covariance matrix for each class for a total of $Kp(p+1)/2$ parameters. If we have 50 predictors this is some multiple of $(50 * 51)/2 = 1275$ for a significant jump in predictors. The LDA model assumes the K classes share a common covariance matrix so that the LDA model becomes linear in x so that there are kp linear coefficients to estimate. Thus LDA is a less flexible classifer than QDA and has a significantly lower variance. The tradeoff comes from noting that if the LDA assumption of a common covariance matrix is incorrect then high bias can be an issue.

We build the Quadratic Discriminant Analysis model, *gda_m1* using the *qda()* function with *mpg_01* as the response variable and *cylinders, displacement, horsepower,* and *weight* as the predictors.

We also output the confusion matrix using the *table* function on the arguments *qda.pred* and the testing data with *mpg01* as the response.

```
qda_m1 <- qda(mpg01 ~ cylinders + displacement
              + horsepower + weight, data = training)
qda.pred <- predict(qda_m1, testing)$class

table(qda.pred, testing$mpg01)
```

```
##
## qda.pred  0  1
##        0 51  5
##        1  7 53
```

1. Compute error for qda model m1 We compute the error to have a measurement of the accuracy of our model.

```
error_qda <- mean(qda.pred != testing$mpg01)
error_qda
```

```
## [1] 0.1034483
```

(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?

Logistic regression models the probability that the response variable belongs to a specific category where $Y = Pr(X)$ where $Pr(X) = Pr(Y = k|X = x)$ is the conditional probability of the response Y given the predictor(s) X. Because this is a probability we need outputs bounded from 0 to 1 for all values of X. In simple logistic regression we use the logistic function for this purpose where

$$Pr(X) = \frac{e^{\beta_0+\beta_1 X_1}}{1 + e^{\beta_0+\beta_1 X_1}}$$

To fit this model we use the maximum likelihood function where we seek to maximize the likelihood function to find the estimates of our predictor coefficients

$$l(\beta_0, \beta_1) = \prod_{i:y_i=1} p(x_i) \prod_{i':y'_i} (1 - p(x'_i))$$

where the estimates $\hat{\beta}_0$ and $\hat{\beta}_1$ are chosen to maximize this likelihood.

We find the *odds* by manipulating $Pr(X)$ to get

$$log\Big(\frac{p(X)}{1 - p(X)}\Big) = \beta_0 + \beta_1 X_1$$

where the left hand side is the logit function or log-odds function.

Multiple logistic regression simply expands upon this model as in multiple linear regression.

$$Pr(X) = \frac{e^{\beta_0+\beta_1 X_1+...+\beta_p X_p}}{1 + e^{\beta_0+\beta_1 X_1+...+\beta_p X_p}}$$

and the logit function can be generalized as

$$log\Big(\frac{p(X)}{1 - p(X)}\Big) = \beta_0 + \beta_1 X_1 + ... + \beta_p X_p$$

And, as with the simple logistic regression, we use the maximum likelihood function to estimate $\beta_0, \beta_1, ..., \beta_p$

We build the Logistic Regression model, *logisticR_m1* using the generalized linear model, *glm()*, function with *mpg_01* as the response variable and *cylinders, displacement, horsepower,* and *weight* as the predictors on the training data using the binomial family for the logit function.

We also output the confusion matrix using the *table* function on the arguments *logisticR_m1* and the testing data with *mpg01* as the response.

```
logisticR_m1 <-glm(mpg01 ~ cylinders + displacement + horsepower + weight, data = training, family = bi

logisticR.prob <- predict(logisticR_m1, testing, type = "response")

logisticR.pred <- ifelse(logisticR.prob > 0.5, 1, 0)
```

```
table(logisticR.pred, testing$mpg01)
```

```
##
## logisticR.pred  0  1
##              0 53  3
##              1  5 55
```

1. compute the error for logistic regression model logisticR_m1. We want the error to have a metric to understand the strength of our model.

```
error_glm <- mean(logisticR.pred != testing$mpg01)
error_glm
```

```
## [1] 0.06896552
```

(g) Perform KNN on the training data, with K = 3, in order to predict mpg01 using the variables that seemed most associated with mpg01 in (b). What test errors do you obtain?

Given a positive integer K and a test observation x_0, the K-nearest neighbors, KNN, classifier identifies the K closest points to x_0, represented by N_0 then estimates the conditional probability for class j as the proportion of points in N_0 whose response values equal j:

$$Pr(Y = j|X = x_0) = \frac{1}{K} \sum_{i \in N_0} I(y_i = j)$$

Then KNN applies Bayes rule and classifies the teset observation x_0 to the class with the largest probability.

KNN can produce classifiers that are very close in predictive ability to the optimal Bayes classifier although the choice of K can have a large effect on the KNN classifier obtained. At $K = 1$ the decision boundary is highly flexible and finds patterns in the boundary that don't match the Bayes decision boundary with low bias but very high variance. With a large K we have less flexibility and a low variance, high bias classifier.

We build our KNN model by using the *cbind()* function to bind the variables of interest together into a matrix for the test and the training data. We then use the *knn()* function on the *test* and *train* matrices with our K value set at 3 and also create a *confusion matrix* to have a better understanding of the model findings.

```
training.X <- cbind(training$cylinders, training$displacement, training$horsepower, training$weight)
testing.X <- cbind(testing$cylinders, testing$displacement, testing$horsepower, testing$weight)

knn.pred <- knn(training.X, testing.X, training$mpg01, k = 3)

table(knn.pred, testing$mpg01)
```

```
##
## knn.pred  0  1
##        0 52  3
##        1  6 55
```

1. compute the error for knn model. The error provides a strength of fit measure for the model that we can compare with our other classifier models.

```
error_knn <- mean(knn.pred != testing$mpg01)
error_knn
```

```
## [1] 0.07758621
```

(h) Perform GAM on the training data in order to predict mpg01 using the variables that seemed most associated with mpg01 in (b). What test error do you obtain?

Generalized Additive Models (GAMs) allow the extension of a standard linear model by allowing non-linear functions of each of the variables while maintaining additivity. GAMs can be used to extend multiple linear regression as well as in situation where Y is a qualitative variable which is what we are concerned with here in the classification setting.

Assume for simplicity that Y takes on the values 0 or 1 and let $p(X) = Pr(Y = 1|X)$ be the conditional probability (given the predictors) that the response equals one.

From the logistic regression model

$$log\Big(\frac{p(X)}{1-p(X)}\Big) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_p X_p$$

This *logit* is the log of the odds of $P(Y = 1|X)$ versus $P(Y - 0|X)$ which represents as a linear function of the predictors. We extend this to use non-linear functions using the model

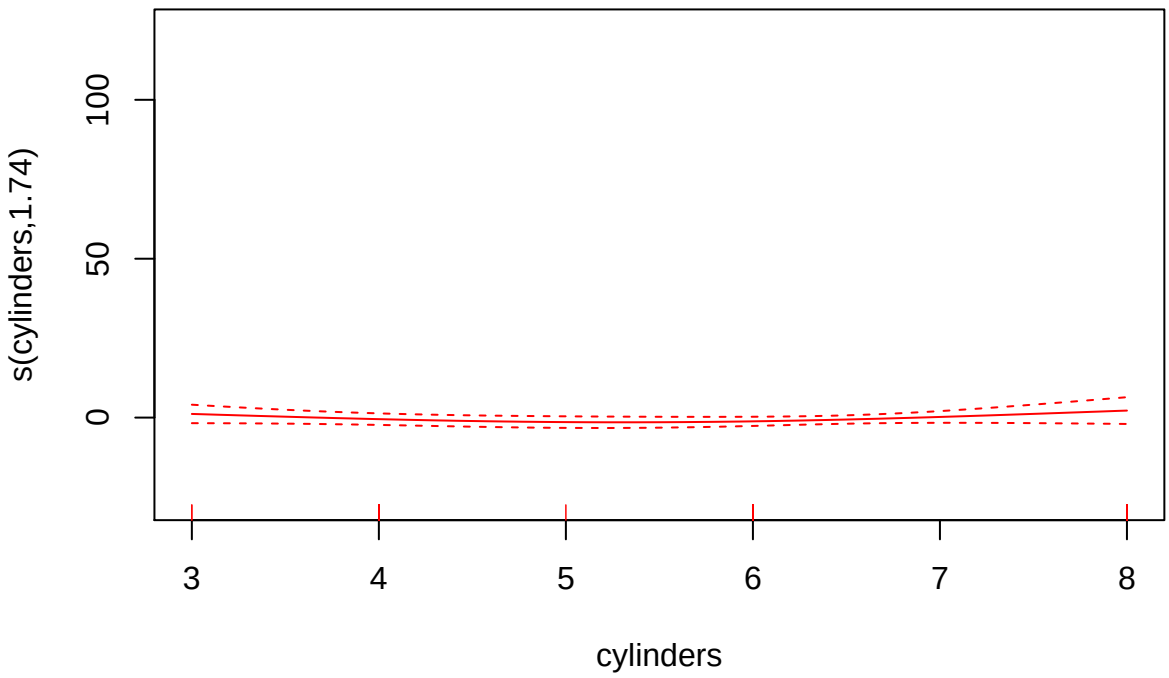
$$log\Big(\frac{p(X)}{1-p(X)}\Big) = \beta_0 + f_1(X_1) + f_2(X_2) + ... + f_p(X_p)$$

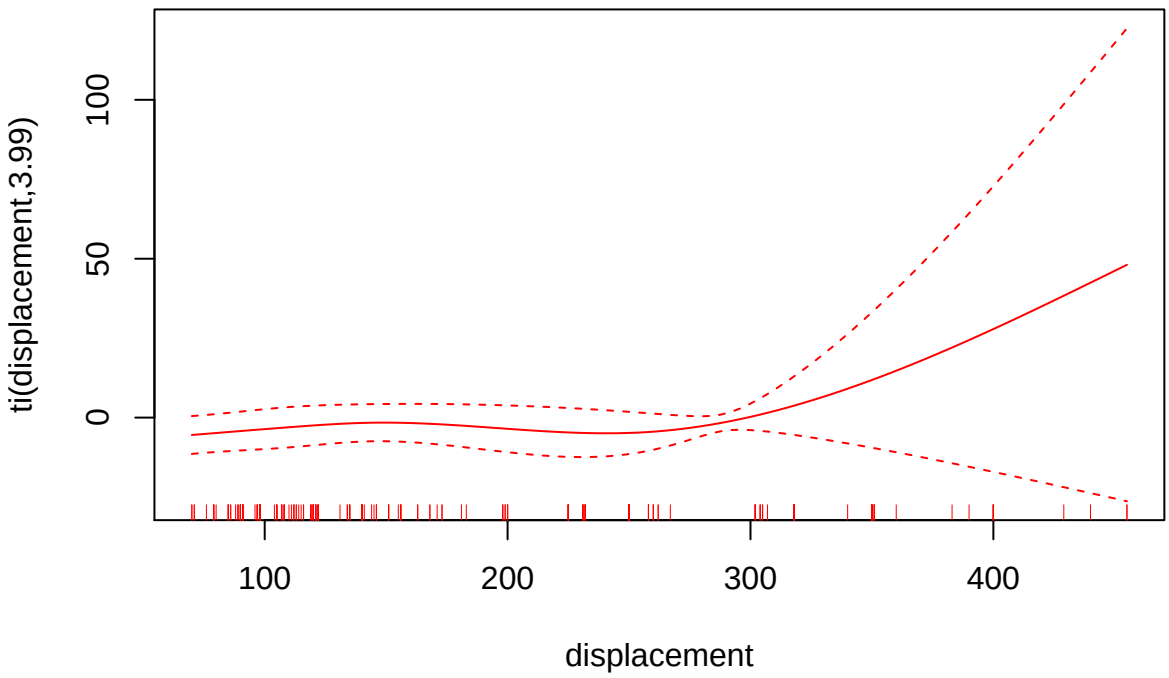
We build the GAM model *gam_m1* using the *gam()* function and using the spline function *s()* on the predictor 'cylinders' and the tensor product functions, *ti()* and *te()* on the interaction terms of interest and using the logit function by selecting the binomial family.

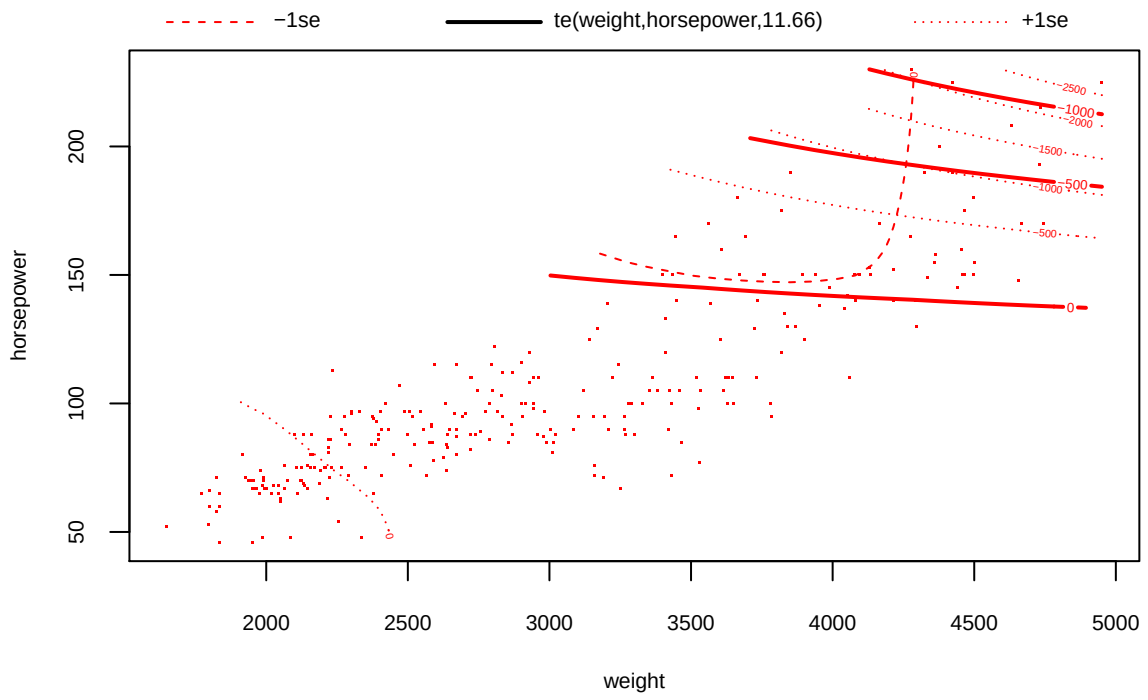
```
gam_m1 <- gam(mpg01 ~ s(cylinders, k = 3) + ti(displacement) + te(weight, horsepower), data = training,

gam_m1.prob <- predict(gam_m1, data = testing, type = "response")

gam_m1.pred <- rep(0, length(gam_m1.prob))

gam_m1.pred[gam_m1.prob > 0.5] <- 1
```

Using the *plot.gam()* function we can visualize the model output.

```
plot.gam(gam_m1, se=TRUE, col='red')
```


s(cylinders,1.74)
100
50
0
3
4
5
6
7
8
cylinders


ti(displacement,3.99)
100
50
0
100
200
300
400
displacement



We can output the error of our model.

```
error_gam <- mean(gam_m1.pred != testing$mpg01)
```

```
## Warning in gam_m1.pred != testing$mpg01: longer object length is not a multiple
## of shorter object length
```

```
error_gam
```

```
## [1] 0.5217391
```