STAT462 Assignment 1

Graham Greig SN: 47022356

S2 2021

1 Problem 1

Describe one advantage and one disadvantage of flexible (versus a less flexible) approaches for regression. Under what conditions might a less flexible approach be preferred?

The main advantage of a flexible model vs. a less flexible model is that it can be used to predict a wider range of possible shapes to fit the known training data. As well, for a hyper-flexible model may be able to directly interpolate the data resulting in near zero training error. A disadvantage of this is that it can lead to overfitting such that when the model is used on test data it has an increased error. If we are mainly interested in inference, a less flexible model is likely a better one. This is because we can infer the trend of the response variable to the predictors with a very simple and easy to compute model. Even a simple linear fit can give us an idea of rising or falling correlation which can be very useful.

2 Problem 2

Consider a binary classification problem $Y \in 0,1$ with one predictor X. The prior probability of being in class 0 is $Pr(Y = 0) = \pi_0 = 0.69$ and the density function for X in class 0 is a standard normal

$$f_0 = Normal(0,1) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}$$

The density function for X in class 1 is also normal but with $\mu = 1$ and $\sigma^2 = 0.5$

$$f_1 = Normal(1,0.5) = \frac{1}{\sqrt{\pi}} e^{-(x-1)^2}$$

a.) Plot $\pi_0 f_0(x)$ and $\pi_1 f_1(x)$

To start, we know the sum of all probabilities must equal 1. With two classes, this means

$$\pi_1 = 0.31$$

Using the following code, we obtain a plot of these distributions.

```
#Define the sequence to visualize the data over.
x = seq(-5,5,length = 100)
```

```
#Define the class probabilities
pi_0 = 0.69
pi_1 = 1 - 0.69

#Define the functions
y_1 = pi_0*dnorm(x)
y_2 = pi_1*dnorm(x,1,sqrt(0.5))

#Plot the curve out.
plot(x,y_1,col = "blue", lwd = 4 ,type  ='l', main = "Plot of pi_0f_0 and pi_1f_1",
     xlab = "x", ylab = "pi_if_i")
#Plot the second curve.
points(x, y_2, col="dark red", lwd = 4, type = 'l')

legend("topright",legend = c("Class 0", "Class 1"),
        col = c("blue","dark red"),lwd = 4,
        text.col = "black", horiz = FALSE)
```

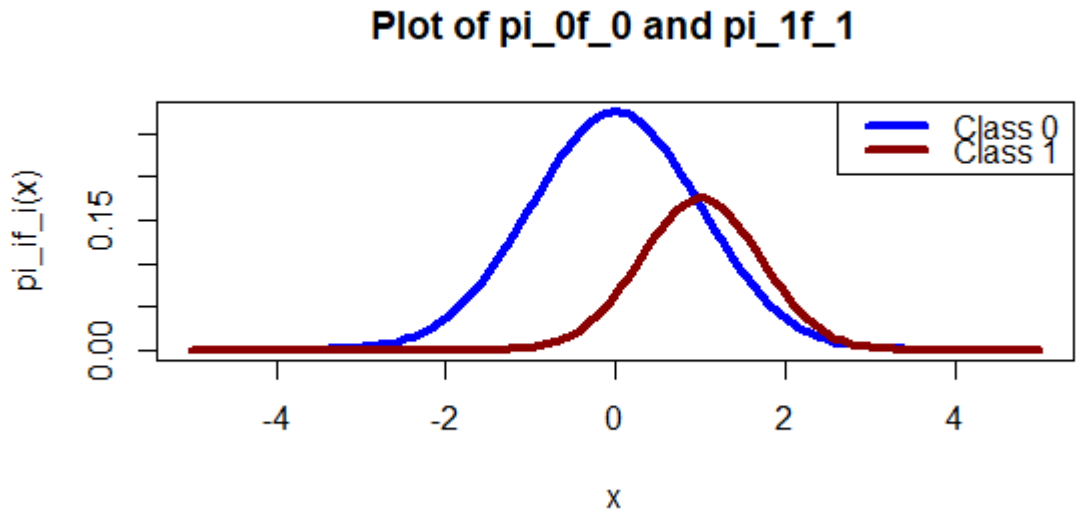

Figure 1: Distributions of Class 1 and Class 0

b.) Find the Bayes decision boundary.

The Bayes decision boundary is found where the probability of each scenario is equally likely. This is found at:

$$\pi_0 f_0(x) = \pi_1 f_1(x)$$

$$\frac{\pi_0}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} = \frac{\pi_1}{\sqrt{\pi}} e^{-(x-1)^2}$$

We can quickly get rid of the $\frac{1}{\sqrt{\pi}}$ on each side and take the natural log to get:

$$ln(\frac{\pi_0}{\sqrt{2}}) - \frac{x^2}{2} = ln(\pi_1) - (x-1)^2$$

$$\frac{x^2}{2} - 2x + (1 + ln(\frac{\pi_0}{\pi_1\sqrt{2}})) = 0$$

Using Wolfram Alpha I plugged in this equation to quickly find the roots to be:

$$root_1 = 0.954577$$

$$root_2 = 3.04542$$

These roots are the decision boundaries. Before root 1 and after root 2, it is most probable to be in class 0 indicating this will be the classifier for these areas. Between root 1 and 2, class 1 is more probable and the bayes classifier chooses this. This is confirmed by the plot from part A.

c.) Using Bayes classifier, classify the observation X = 3.

Given that X = 3 lies between roots 1 and 2, and the probability of X being in class 1 is higher in this region, the Bayes classifier will choose class 1. It is hard to confirm this from looking at the plot as 3 is very close to

the second root. However, we can simply calculate the probability of X = 3 in class 1 to confirm this.

$$Pr(Y = 1|X = 3) = \frac{\pi_1 f_1(3)}{\pi_1 f_1(3) + \pi_0 f_0(3)}$$

Plugging the equations into Wolfram Alpha at X = 3 gives:

$$Pr(Y = 1|X = 3) = 0.5116$$

Since this is greater than 0.5, class 1 is chosen by the Bayes classifier.

d.) What is the probability that an observation with X = 2 is in class 1?

Using the same formula as in part c, we can calculate the probability at X = 2. This is again solved using Wolfram Alpha.

$$Pr(Y = 1|X = 2) = \frac{\pi_1 f_1(2)}{\pi_1 f_1(2) + \pi_0 f_0(2)} = 0.63331$$

3 Problem 3

In this question, you will fit kNN regression models to the Auto data set to predict Y = mpg using X = horsepower. This data has been divided into training and testing sets. The kNN R function on lean will be used to answer this question.

a.) Perform kNN regression with k = 2,5,10,20,30,50 and 100. Compute the training and testing MSE for each value of k.

To perform this question the kNN regression needs to be run twice. Once with x.pred = x.train and once with x.pred = x.test. This is done to make sure that the x positions are properly aligned for each and the MSE can be compared. Using:

$$k = 2, 5, 10, 20, 30, 50, 100$$

The training and testing MSE are found to be:

$$MSE_{Tr} = 11.67, 15.40, 17.38, 17.49, 19.00, 19.48, 26.26$$

$$MSE_{Te} = 22.86, 19.56, 18.63, 17.32, 17.93, 19.57, 26.32$$

The code for this problem is shown here:

```
#####################################################
# Problem 3
# Code given for kNN
## STAT318/462 kNN regression function

kNN <- function(k,x.train,y.train,x.pred) {
  #
  ## This is kNN regression function for problems with
  ## 1 predictor
  #
  ## INPUTS
  #
  # k       = number of observations in nieghbourhood
  # x.train = vector of training predictor values
  # y.train = vector of training response values
  # x.pred  = vector of predictor inputs with unknown
  #           response values
  #
  ## OUTPUT
  #
  # y.pred  = predicted response values for x.pred
```

```
  ## Initialize:
  n.pred <- length(x.pred); y.pred <- numeric(n.pred)

  ## Main Loop
  for (i in 1:n.pred){
    d <- abs(x.train - x.pred[i])
    dstar = d[order(d)[k]]
    y.pred[i] <- mean(y.train[d <= dstar])
  }
  ## Return the vector of predictions
  invisible(y.pred)
}

# Load in the data
AutoTrain=read.csv("AutoTrain.csv",header=T,na.strings="?")
AutoTrain = AutoTrain[order(AutoTrain$horsepower),]
AutoTest=read.csv("AutoTest.csv",header=T,na.strings="?")
AutoTest = AutoTest[order(AutoTest$horsepower),]

# Get variables of interest
x.train = AutoTrain$horsepower
y.train = AutoTrain$mpg
x.test = AutoTest$horsepower
y.test = AutoTest$mpg

# Setup the loop variables
k = c(2,5,10,20,30,50,100)
MSE.Train = numeric(length(k))
MSE.Test = numeric(length(k))

#Loop over every value of k, train the data and compute the MSE
```

```
# This is run twice to ensure the data has the proper X positions.
for (i in 1:length(k)){
  y.pred_train = kNN(k[i],x.train,y.train,x.train)
  y.pred_test = kNN(k[i],x.train,y.train,x.test)
  MSE.Train[i] = mean((y.train - y.pred_train)^2)
  MSE.Test[i] = mean((y.test - y.pred_test)^2)
}
```

b.) Which value of k performed best?

We expect the mean squared testing error to decrease as k decreases until it reaches or gets close to the irreducible error. Whereas the mean squared testing error will continue to decrease as k decreases. The following plot shows this.

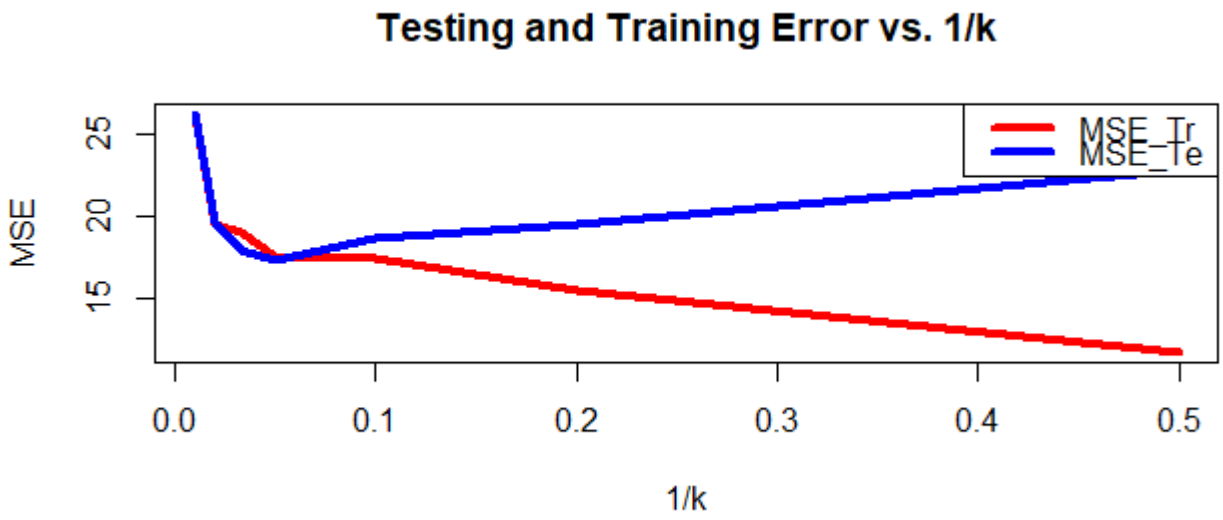

Figure 2: Training and testing mean squared error.

The bounce point of the MSE_{Te} is where the model performs best for the training data and this is found to be at $k = 20$.

code:

```
plot(1/k,MSE.Train,col = 'red', lwd = 4, type = 'l', main = 'Testing and Training Error vs. 1/k', xlab =
points(1/k,MSE.Test,col = 'blue', lwd = 4, type = 'l')
```

```
legend("topright",legend = c("MSE_Tr", "MSE_Te"),
       col = c("red","blue"),lwd = 4,
       text.col = "black",
       horiz = FALSE)
```

c.) Plot the training data, testing data and the best kNN model.

Plotting the training, testing and best prediction gives the following plot.

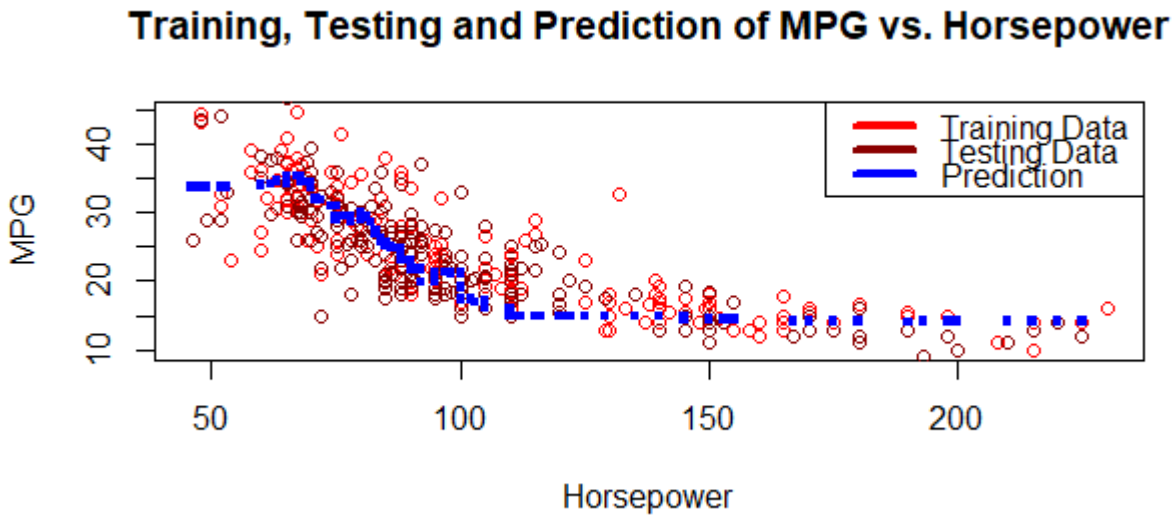

Figure 3: Training Data, Testing Data and Prediction with $k = 20$

code:

```
y.pred = kNN(20,x.train,y.train,x.train)
plot(x.train,y.train,col = "red" , main = "Training, Testing and Prediction of MPG vs. Horsepower",
     xlab = "Horsepower", ylab = "MPG")
points(x.test,y.test,col = "dark red")
points(x.test,y.pred,pch = 15, col = "blue", cex = 0.6)
legend("topright",legend = c("Training Data", "Testing Data", "Prediction" ),
       col = c("red","dark red",'blue'),lwd = 4,
       text.col = "black",
       horiz = FALSE)
```

d.) Describe the bias-variance trade off for kNN regression.

The bias-variance trade-off helps describe the U-shaped distribution of the MSE_{Te} curve as compared to the steadily decreasing MSE_{Tr} curve vs $\frac{1}{k}$. The bias-variance trade-off states that as the flexibility of the model increases the variance of the predictor model increases and the bias decreases.As the flexibility decreases, the opposite takes place with the bias increasing and the variance decreasing. In a kNN regression model, the flexibility is controlled by k. Lower values of k will have a higher variance as they tend to fit very different functions with small changes in the training data set. This leads to over-fitting of the training data, a low MSE_{Tr} and a higher MSE_{Te}. Higher values of k have a lower variance but tend to pull average values across much of the data set which leads to under-fitting of the data. This under-fitting is a bias of to the regression model and leads to a high MSE_{Te} and MSE_{Tr}. The point of best prediction is found at the bottom of the U shaped distribution of the testing data where the bias and variance are minimized.