Rhys Jennings - rpj19 29656157

Rhys Jennings - STAT318 - Assignment 1

17/03/2020

1)

- Advantage: A **less flexible** approach will mean that we may be underfitting the data, which means we may not be explaining the data very accurately. For example, with very low flexibility we will only be explaining the data with a near-linear model, which may be inaccurate. So if we need to explain a more complicated relationship, we would want to use a more flexible method.
- Disadvantage: If we have a **high flexibility**, this will mean that we will likely be overfitting the data, which means that we might be trying to explain random errors which is not helpful when trying to find a model for the whole population.

Conditions to prefer **high flexibility**:

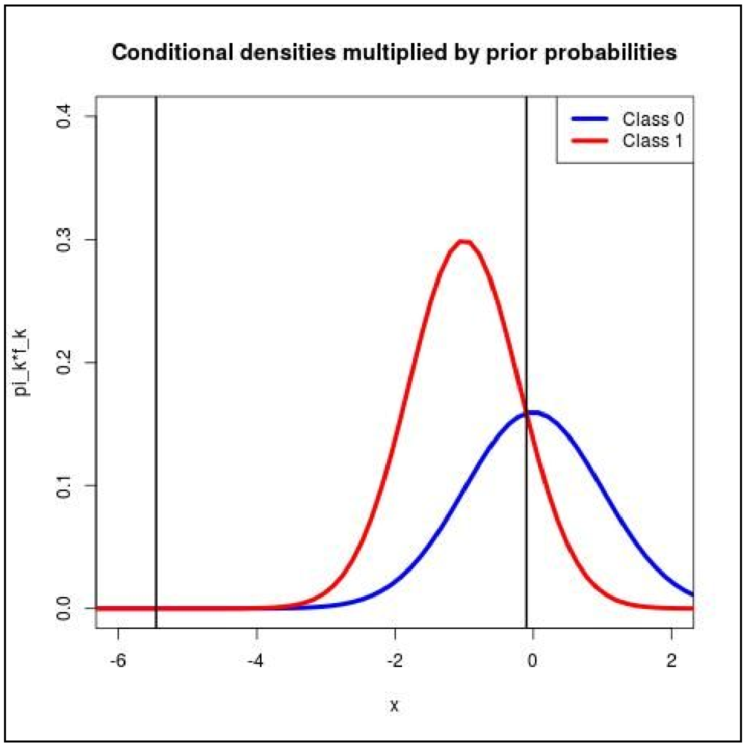
- If we want to reduce the bias then we would use a more flexible approach.
- If the relationship between the predictors and the response is non-linear, then we would require a more flexible model to accurately describe the data.

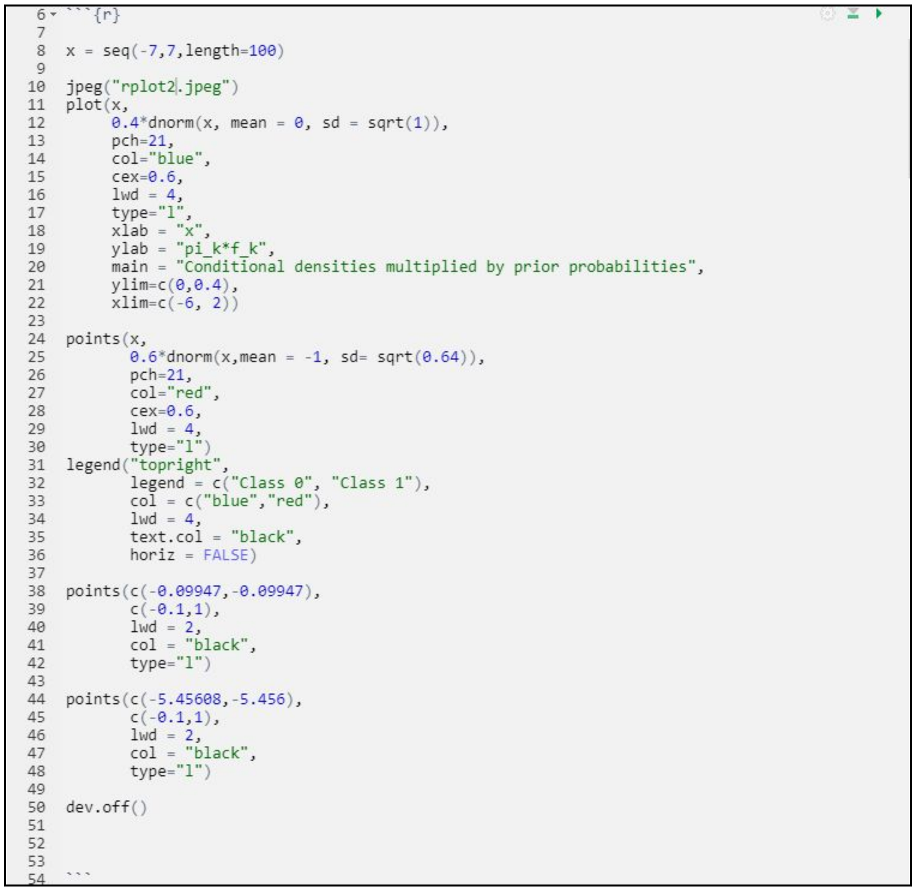
2) a)b)

In order to find the boundaries we solve both density functions to find x.

- ***The full worked solution on solving this equation is on the last page***

This results in the boundaries x = (-5.45608.., -0.09946..)

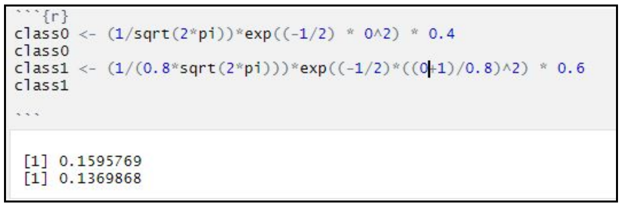




~~~
```{r}

x = seq(-7,7,length=100)

jpeg("rplot2.jpeg")
plot(x,
     0.4*dnorm(x, mean = 0, sd = sqrt(1)),
     pch=21,
     col="blue",
     cex=0.6,
     lwd = 4,
     type="l",
     xlab = "x",
     ylab = "pi_k*f_k",
     main = "Conditional densities multiplied by prior probabilities",
     ylim=c(0,0.4),
     xlim=c(-6, 2))

points(x,
       0.6*dnorm(x,mean = -1, sd= sqrt(0.64)),
       pch=21,
       col="red",
       cex=0.6,
       lwd = 4,
       type="l")
legend("topright",
       legend = c("Class 0", "Class 1"),
       col = c("blue","red"),
       lwd = 4,
       text.col = "black",
       horiz = FALSE)

points(c(-0.09947,-0.09947),
       c(-0.1,1),
       lwd = 2,
       col = "black",
       type="l")

points(c(-5.45608,-5.456),
       c(-0.1,1),
       lwd = 2,
       col = "black",
       type="l")

dev.off()



```
~~~

This code results in creating the graph with the correct distributions multiplied by their probabilities. It also includes lines showing where the boundaries are

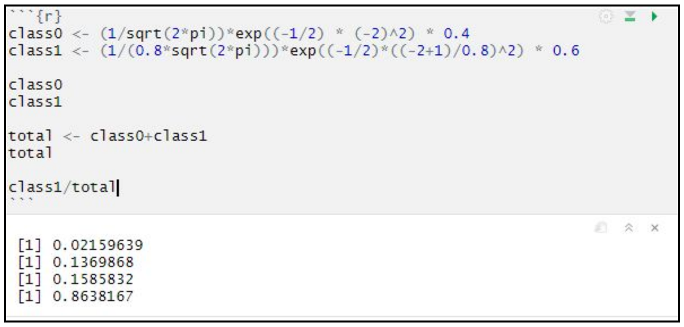
c)



~~~
```{r}
class0 <- (1/sqrt(2*pi))*exp((-1/2) * 0^2) * 0.4
class0
class1 <- (1/(0.8*sqrt(2*pi)))*exp((-1/2)*((0+1)/0.8)^2) * 0.6
class1

```

[1] 0.1595769
[1] 0.1369868
~~~

- After putting the value 0 into both density functions, we can see that the value 0 is more likely to be in class 0 rather than class 1 as our bayes classifier predictor value is higher for class 0.

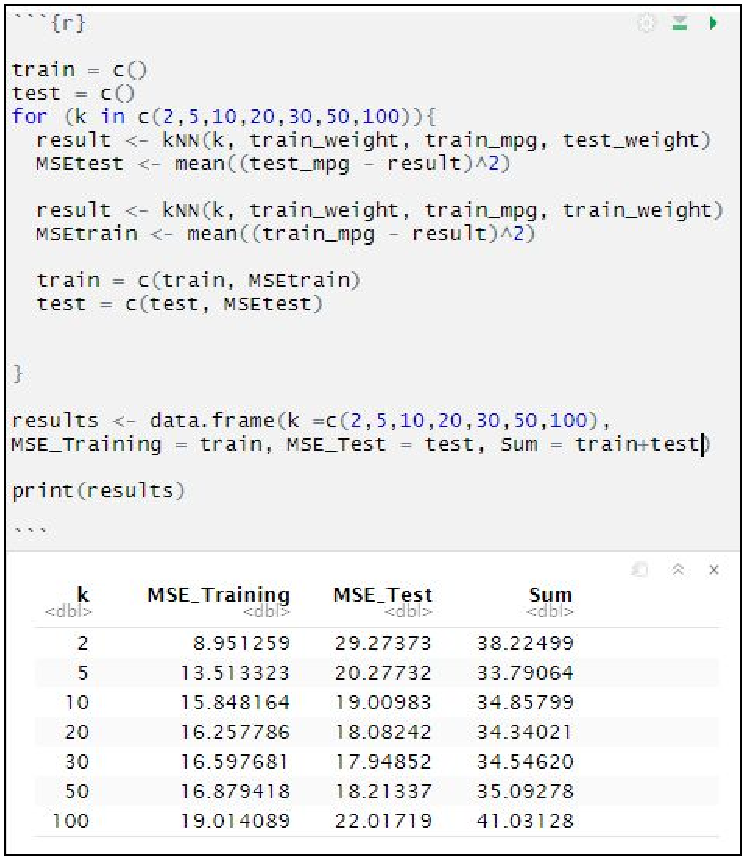
d)



~~~
```{r}
class0 <- (1/sqrt(2*pi))*exp((-1/2) * (-2)^2) * 0.4
class1 <- (1/(0.8*sqrt(2*pi)))*exp((-1/2)*((-2+1)/0.8)^2) * 0.6

class0
class1

total <- class0+class1
total

class1/total
```

[1] 0.02159639
[1] 0.1369868
[1] 0.1585832
[1] 0.8638167
~~~

- Once we find the predictors values for each classifier, we can see what the probability X has of being from each classifier. In this case (X=-2), we found the predictor value for class0 to be 0.02(2dp) and for class1 to be 0.14(2dp). This sums to give 0.16(2dp), in which we can then divide the predictor from class1 by the total, which results in a 86.38% (2dp) chance of being from class1.

3)a)



~~~
```{r}

train = c()
test = c()
for (k in c(2,5,10,20,30,50,100)){
  result <- kNN(k, train_weight, train_mpg, test_weight)
  MSEtest <- mean((test_mpg - result)^2)

  result <- kNN(k, train_weight, train_mpg, train_weight)
  MSEtrain <- mean((train_mpg - result)^2)

  train = c(train, MSEtrain)
  test = c(test, MSEtest)


}

results <- data.frame(k =c(2,5,10,20,30,50,100),
MSE_Training = train, MSE_Test = test, Sum = train+test)

print(results)

```
~~~

```
    k   MSE_Training   MSE_Test        Sum
<dbl>          <dbl>      <dbl>      <dbl>
    2       8.951259   29.27373   38.22499
    5      13.513323   20.27732   33.79064
   10      15.848164   19.00983   34.85799
   20      16.257786   18.08242   34.34021
   30      16.597681   17.94852   34.54620
   50      16.879418   18.21337   35.09278
  100      19.014089   22.01719   41.03128
```

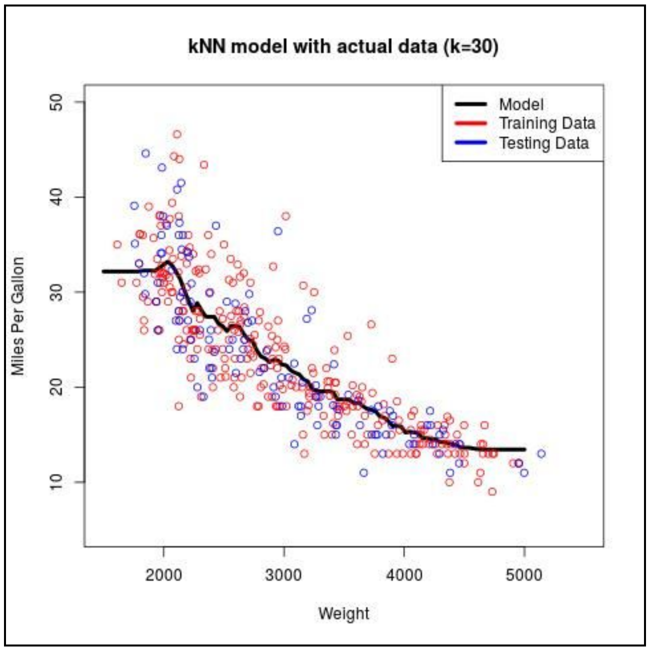
Here I am looping through every value of k we would like to try, and computing the MSE for the training data and the MSE for the test data after using the kNN function for each k.

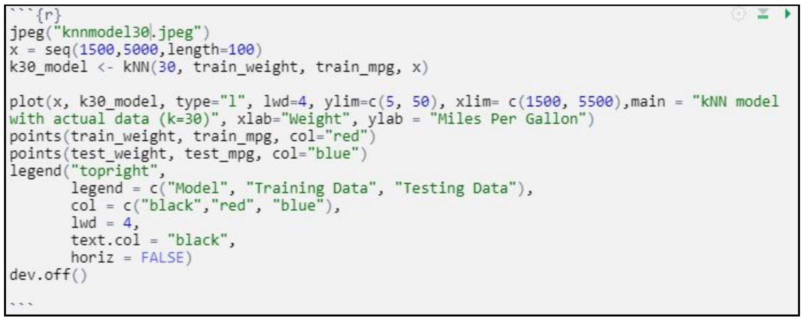
I then put all values into a dataframe for easy viewing.

b)

- We can see that a k of 30 produces a low MSE for the training data, and also the lowest MSE for the test, meaning we are not trying to explain patterns in the training data that are not apparent in the test data. A k of 30 and 10 also perform similarly well but a k of 30 seems to allow us to get a reasonably low MSE for both the training and the test data.

c)





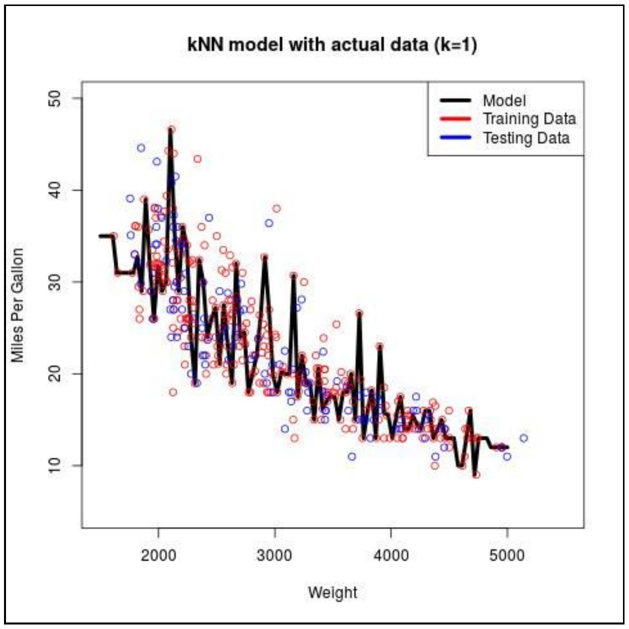
~~~
```{r}
jpeg("knnmodel30.jpeg")
x = seq(1500,5000,length=100)
k30_model <- kNN(30, train_weight, train_mpg, x)

plot(x, k30_model, type="l", lwd=4, ylim=c(5, 50), xlim= c(1500, 5500),main = "kNN model
with actual data (k=30)", xlab="Weight", ylab = "Miles Per Gallon")
points(train_weight, train_mpg, col="red")
points(test_weight, test_mpg, col="blue")
legend("topright",
       legend = c("Model", "Training Data", "Testing Data"),
       col = c("black","red", "blue"),
       lwd = 4,
       text.col = "black",
       horiz = FALSE)
dev.off()

```
~~~

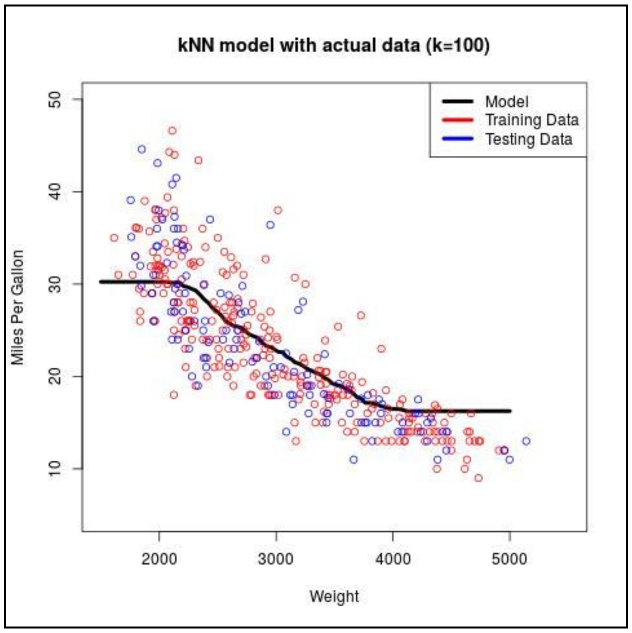
- We can see that our kNN model gives a relatively suitable fit for all the data we have. There is a lot of variability at the beginning of the model, so this is the area in which the model fits the data the least, however as the weight increases, our model fits the data increasingly well.

d)

- The variance means the amount our model would change if we estimated it using different training data. Ideally, we wouldn't want the model to change much using different data sets, as we want it to be an estimate for the entire population. Using a small k would result in an over-fitted line, which only explains the data in the training set. We can see this in the graph below, with a k of 1.

- Bias occurs when we are simplifying real data, meaning that we are likely over-simplifying the model, which would mean our model would be less accurate at explaining the whole population. For example, a very high k would result in an under-fitted line which may not sufficiently explain patterns in the data. We can see this in the picture below with a k of 100. However a large k decreases variance.





- The bias-variance trade off is when we need to find a balance between having two much variance, or too much bias on our model. A higher k results in more variance (less bias), while a lower k results in more bias (and less variance). This is why we needed to find an effective k for defining a neighbourhood.

- Starting equation

$$0.4 * \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2} = 0.6 * \frac{1}{0.8\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{x+1}{0.8})^2}$$

- Moving numbers to one side

$$\frac{0.8\sqrt{2\pi}0.4}{0.6\sqrt{2\pi}} e^{\frac{-x^2}{2}} = e^{-\frac{1}{2}(\frac{x+1}{0.8})}$$

- Simplifying the left side

$$\frac{0.32}{0.6} e^{\frac{-x^2}{2}} = e^{-\frac{1}{2}(\frac{x+1}{0.8})}$$

- Applying log rules to both sides of the equation

$$ln(\frac{0.32}{0.6}) - \frac{1}{2}x^2 * ln(e) = -\frac{1}{2}(\frac{x+1}{0.8})^2 * ln(e)$$

- Removing ln(e)

$$2ln(\frac{0.32}{0.6}) - x^2 = -(\frac{x+1}{0.8})^2$$

- Expanding the right side and adding x^2 to both sides

$$2ln(\frac{0.32}{0.6}) = -\frac{x^2}{0.64} - \frac{2x}{0.64} - \frac{1}{0.64} + x^2$$

- Multiplying both sides by 0.64

$$-x^2 - 2x - 1 + 0.64x^2 = 0.64 * 2ln(\frac{0.32}{0.6})$$

- Simplifying the equation

$$-0.36^2 - 2x - 0.19538.. = 0$$

- Applying the quadratic equation.

$$x = \frac{2 \pm \sqrt{-2^2 - 4 * -0.36 * -0.1953..}}{2 * -0.36}$$

<u>This results in the boundaries x = -5.456.., -0.099..</u>