FIT2086 Assignment 3

Question 1

1.1 Use cross-validation with 10 folds and 1000 repetitions to select an appropriate size tree. What variables have been used in the best tree? How many leaves (terminal nodes) does the best tree have?

```
 heart.train1=rpart(HD ~ ., heart.train)
 # There are 7 terminal nodes in this tree
 # The variables are CA, CP, EXANG, CHOL, AGE
 heart.train1
= 210

ode), split, n, loss, yval, (yprob)
      * denotes terminal node

1) root 210 96 N (0.5428571 0.4571429)
  2) THAL=Normal 114 25 N (0.7807018 0.2192982)
    4) AGE< 54.5 64  4 N (0.9375000 0.0625000) *
    5) AGE>=54.5 50 21 N (0.5800000 0.4200000)
     10) EXANG=N 36 11 N (0.6944444 0.3055556)
       20) CHOL< 304 28  6 N (0.7857143 0.2142857) *
       21) CHOL>=304 8  3 Y (0.3750000 0.6250000) *
     11) EXANG=Y 14  4 Y (0.2857143 0.7142857) *
  3) THAL=Fixed.Defect,Reversible.Defect 96 25 Y (0.2604167 0.7395833)
    6) CP=Atypical,NonAnginal,Typical 30 13 N (0.5666667 0.4333333)
     12) CA< 0.5 20  6 N (0.7000000 0.3000000) *
     13) CA>=0.5 10  3 Y (0.3000000 0.7000000) *
    7) CP=Asymptomatic 66  8 Y (0.1212121 0.8787879) *
```

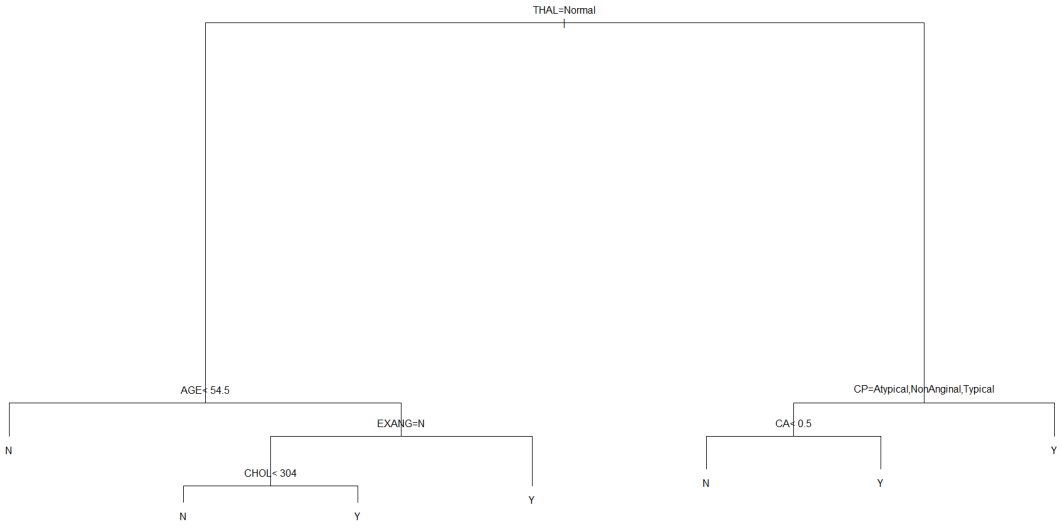
Answer: As it can see above, there are 7 terminal nodes in the best tree and the variables are CA, CP, EXANG, CHOL, AGE, THAL.

1.2 Plot the tree found by CV and discuss about the relationship between the predictors and heart disease.

The R-code below shows how to plot the best tree by CV:

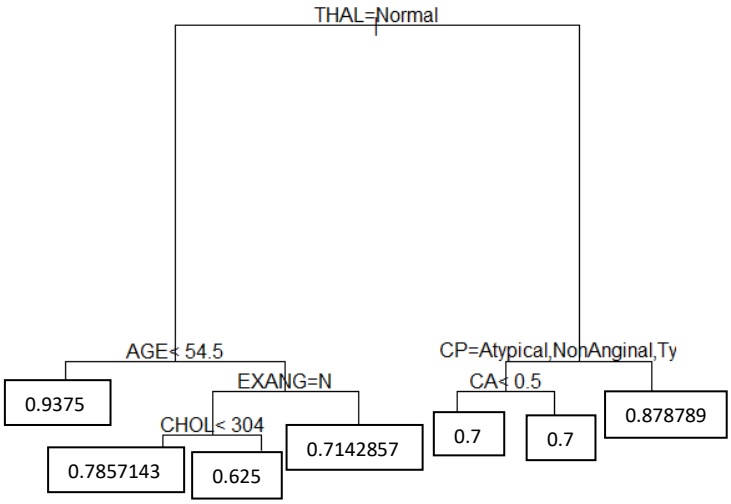
```
> #1.2
> cv = learn.tree.cv(HD~.,data=heart.train,nfolds=10,m=1000)
> plot.tree.cv(cv)
> prune.rpart(tree=heart.train1, cp = cv$best.cp)
n= 210

node), split, n, loss, yval, (yprob)
      * denotes terminal node

 1) root 210 96 N (0.5428571 0.4571429)
   2) THAL=Normal 114 25 N (0.7807018 0.2192982)
     4) AGE< 54.5 64  4 N (0.9375000 0.0625000) *
     5) AGE>=54.5 50 21 N (0.5800000 0.4200000)
      10) EXANG=N 36 11 N (0.6944444 0.3055556)
        20) CHOL< 304 28  6 N (0.7857143 0.2142857) *
        21) CHOL>=304 8  3 Y (0.3750000 0.6250000) *
      11) EXANG=Y 14  4 Y (0.2857143 0.7142857) *
   3) THAL=Fixed.Defect,Reversible.Defect 96 25 Y (0.2604167 0.7395833)
     6) CP=Atypical,NonAnginal,Typical 30 13 N (0.5666667 0.4333333)
      12) CA< 0.5 20  6 N (0.7000000 0.3000000) *
      13) CA>=0.5 10  3 Y (0.3000000 0.7000000) *
     7) CP=Asymptomatic 66  8 Y (0.1212121 0.8787879) *
> plot(cv$best.tree)
> text(cv$best.tree, pretty=12)
```

The tree below shows the relationship between the predictors and the heart disease:



Answer: The tree above gives a relationship between the predictors and heart disease is that the tree can be split on the same numeric variable for more than once. The tree will either be split to the left-hand subtree or right-hand subtree. Therefore, by knowing the THAL (Thallium scanning result), age, EXANG, CHOL (serum cholesterol), CP (Chest pain type) and CA (Number of major vessels coloured by fluoroscopy) can determine if the person has heart disease or not. For example, if a person THAL=Normal, Age<54.5, it will move to the left-hand subtree at the first split, then the right-hand branch and arrive at the leaf with a predicted chance of NO for getting heart disease. If a person THAL=Normal, AGE>54.5, EXANG=N, CHOL<304, it will move to the left-hand subtree at the first split, then the right-hand branch, then the left-hand branch, then the left-hand branch and arrive at the leaf with a predicted chance of NO for getting heart disease. If a person THAL=Normal, AGE>54.5, EXANG=N, CHOL>304, it will move to the left-hand subtree at the first split, then the right-hand branch, then the left-hand branch, then the right-hand branch and arrive at the leaf with a predicted chance of YES for getting heart disease. If a person THAL=Normal, AGE>54.5, EXANG=Y, it will move to the left-hand subtree at the first split, then the right-hand branch, then the right-hand branch and arrive at the leaf with a predicted chance of YES for getting heart disease. If a person THAL is not Normal, CP=Atypical or Non Anginal or Typical and CA <0.5, it will move to the right-hand subtree at the first split, then the left-hand branch, then the left-hand branch and arrive at the leaf with a predicted chance of NO for getting heart disease. If a person THAL is not Normal ,CP=Atypical or Non Anginal or Typical and CA >0.5, it will move to the right-hand subtree at the first split, then the left-hand branch, then the right-hand branch and arrive at the leaf with a predicted chance of YES for getting heart disease. If a person THAL is not Normal, CP=Asymptomatic, it will move to the right-hand subtree at the first split, then the right-hand branch and arrive at the leaf with a predicted chance of YES for getting heart disease. Therefore, this tree diagram has showed the relationship between the predictors and heart disease as all the predictors are related to the chances of getting heart disease.

1.3 Determine the probabilities of having heart disease



1.4 According to your tree, which predictor combination results in the highest probability of having heart-disease?

Answer: According to the tree, the predictor combination of when THAL is not Normal and CP=Asymptomatic has the highest probability of having heart-disease because the predicted values are the highest compared to others. The predicted value is 0.878789 which is the highest probability of having heart-disease according to the tree. The reason why the predicted value 0.9375 is not selected because although it has the highest probability value, but the probability value is when the person does not have heart-disease which result in highest probability of **not** having heart-disease. Therefore, the predictor combination of THAL is not Normal and CP=Asymptomatic has the highest probability of having heart-disease.

1.5 What variables does the final model include, and how do they compare with the variables used by the tree estimated by CV? Which predictor is the most important in the logistic regression?

```
> #1.5
> file= glm(HD ~ . , data=heart.train, family=binomial)
> step.fit.bic = step(file, direction="both", k=log(nrow(heart.train)), trace=0)
> summary(step.fit.bic)

Call:
glm(formula = HD ~ CP + EXANG + OLDPEAK + CA + THAL, family = binomial,
    data = heart.train)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.3443  -0.5519  -0.2818   0.4447   2.5480

Coefficients:
                      Estimate Std. Error z value Pr(>|z|)
(Intercept)            -0.8075     0.7810  -1.034 0.301145
CPAtypical             -0.8793     0.6180  -1.423 0.154768
CPNonAnginal           -1.4017     0.4975  -2.817 0.004844 **
CPTypical              -2.6324     0.8937  -2.945 0.003225 **
EXANGY                  1.3348     0.4569   2.922 0.003483 **
OLDPEAK                 0.5519     0.2208   2.499 0.012440 *
CA                      0.8487     0.2553   3.324 0.000888 ***
THALNormal             -0.9973     0.7303  -1.366 0.172065
THALReversible.Defect   0.8879     0.7527   1.180 0.238132
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 289.58  on 209  degrees of freedom
Residual deviance: 156.62  on 201  degrees of freedom
AIC: 174.62

Number of Fisher Scoring iterations: 5
```

Answer: According to the summary step.fit.bic above, the variables that's included are CP, EXANG, OLDPEAK, CA and THAL. The difference between the variables used by the logistic regression model estimated by stepwise selection with BIC score and the variables used by the tree estimated by CV is that there's one extra variable AGE when it's used by the tree estimated by CV. There are 6 variables when it's used by the tree estimated by CV which is CA, CP, EXANG, CHOL, AGE, THAL. Therefore, the difference will be the extra variable AGE. The predictor CA is the most important in the logistic regression as it has 4 * compared to other predictors and the Pr(> |z|) value is the lowest.

1.6 Regression equation for the logistic regression model you found using stepwise selection.

```
glm(formula = HD ~ CP + EXANG + OLDPEAK + CA + THAL, family = binomial,
    data = heart.train)
```

```
> #1.6
> step.fit.bic$coefficients
          (Intercept)           CPAtypical         CPNonAnginal             CPTypical               EXANGY
           -0.8075472           -0.8793322           -1.4017135           -2.6324294            1.3348209
              OLDPEAK                   CA           THALNormal THALReversible.Defect
            0.5519443            0.8486996           -0.9972506            0.8878904
```

By applying the coefficient values above into the formula, the formula will be:

Formula = -0.8075472 + (-0.8793322) *CPAtypical + (-1.4017135) *CPNonAnginal + (-2.6324294) *CPTypical + (1.3348209) *EXANGY + (0.5519443) *OLDPEAK + (0.8486996) *CA + (-0.9972506) *THALNormal + (0.8878904) *THALReversible.Defect

Answer: Formula = -0.8075472 + (-0.8793322) *CPAtypical + (-1.4017135) *CPNonAnginal + (-2.6324294) *CPTypical + (1.3348209) *EXANGY + (0.5519443) *OLDPEAK + (0.8486996) *CA + (-0.9972506) *THALNormal + (0.8878904) *THALReversible.Defect

1.7 Compute the prediction statistics for both the tree and the stepwise logistic regression model on the heart data.

The R-code below is computing the prediction statistics for stepwise logistic regression model:

```
> #1.7
> my.pred.stats(predict(step.fit.bic,heart.train,type="response"), heart.train$HD)
-----------------------------------------------------------------------
Performance statistics:

Confusion matrix:

     target
pred   N   Y
   N 101  20
   Y  13  76

Classification accuracy = 0.8428571
Sensitivity             = 0.7916667
Specificity             = 0.8859649
Area-under-curve        = 0.9063414
Logarithmic loss        = 78.31014
```

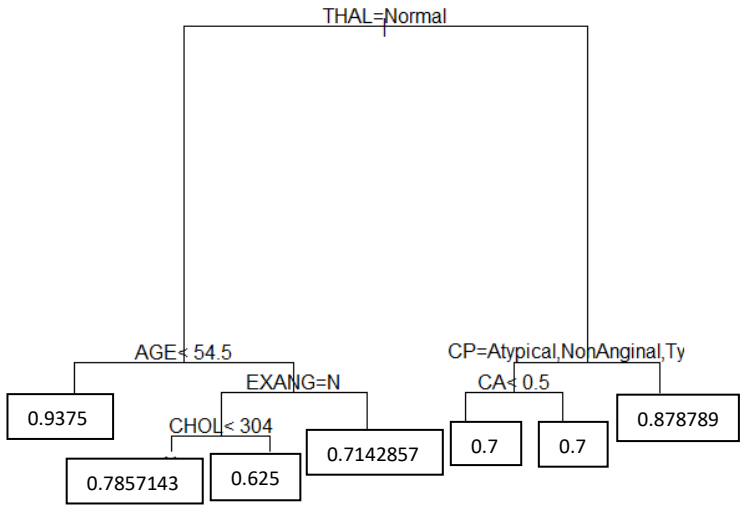
The R-code below is computing the prediction statistics for tree model:

```
-----------------------------------------------------------------------
> my.pred.stats(predict(cv$best.tree,heart.train)[,2], heart.train$HD)
-----------------------------------------------------------------------
Performance statistics:

Confusion matrix:

     target
pred  N  Y
   N 96 16
   Y 18 80

Classification accuracy = 0.8380952
Sensitivity             = 0.8333333
Specificity             = 0.8421053
Area-under-curve        = 0.885508
Logarithmic loss        = 85.8811
```

Answer: Classification accuracy is the percentage of times our model correctly classifies an individual/object. The classification accuracy is the most straightforward measure for performance. This ranges from 0 which is perfectly incorrect to 0.5 which is only as good as random guessing and finally to 1 which is perfectly correct. In this case, the classification accuracy for stepwise logistic regression model is 0.8428571 which is better than the classification accuracy for the tree model which is 0.8380952 because the value is closer to which means that the classification accuracy for stepwise logistic regression model is closer to perfectly correct compared to for the tree model. Sensitivity is the proportion of classifications of individuals as “successes” (or a “1”) that is correct. Sensitivity of 1 means that we correctly classify all individuals for which $y_i' = 1$. In this case, the Sensitivity for the tree model is higher than by using the stepwise logistic regression model as the tree model has the value of 0.8333333 compared to the stepwise logistic regression model which only has 0.7916667.This means that the tree model has a better sensitivity and also better at classifying all individuals correctly compared to stepwise logistic regression model as the value is closer to 1. Specificity is the proportion of classifications of individuals as “failures” (or a “0”) that is correct. Specificity of 1 means we correctly classify all individuals for which $y_i' = 0$. In this case, the specificity for the stepwise logistic model is higher which is 0.8859649 compared to the tree model which is 0.8421053. The value that is closer to 1 means that we correctly classify all individuals for

which $y_i' = 0$. Therefore, stepwise logistic model has a better specificity compared to the tree model. For area-under-the-curve, the bigger the area, the better the classifier. The area under the curve as known as AUC is always between 0 and 1. AUC of 1 means that we can achieve perfect classification, AUC of 0 means that we can achieve perfect misclassification and AUC of 0.5 means that we do no better than a random guess. In this case, the area under the curve is higher in the stepwise logistic regression model compared to the tree model. The value in stepwise logistic regression model is better and closer to achieving the perfect classification as the value is 0.9063414 which is closer to the value of 1 compared to the tree model only has the value of 0.8885508. Lastly, logarithmic loss measures how well the model predicts the probabilities of an individual being in a class. Therefore, the smaller the score, the better our classifier predicts the data. By using stepwise logistic regression model has a lower score which is 78.31014 compared to using the tree model which has 85.8811. This proves that it will be better for the classifier to predict the data by using stepwise logistic regression model as it has smaller score compared to the tree model. In conclusion, although the stepwise logistic regression model has a lower sensitivity compared to the tree model but the classification accuracy, specificity, area-under-the-curve and the logarithmic loss is better compared to the tree model. Therefore, the stepwise logistic regression model will be preferable as a diagnostic test than the tree model because it has a better classification accuracy, specificity, area-under-the-curve and the logarithmic loss which improves the classifier in predicting the data.

1.8 Calculate the odds of having heart disease for the patient in the 45th row of this new dataset for the tree model found using cross-validation and the stepwise logistic regression model.



	AGE	SEX	CP	TRESTBPS	CHOL	FBS	RESTECG	THALACH	EXANG	OLDPEAK	SLOPE	CA	THAL	HD
45	68	M	NonAnginal	180	274	>120	Hypertrophy	150	Y	1.6	Flat	0	Reversible.Defect	Y

By following the tree model and applying all the information for the 45th row above, the probability of having heart disease is 0.30 because the 0.70 is the probability of not having heart-disease. Therefore, 1- 0.70 = 0.30.

By applying the formula odd $= \frac{p}{1-p}$:

The odds of having heart-disease by using the tree model found using cross-validation:

$$\frac{0.3}{(1-0.3)} = 0.42857143$$

```
> #1.8
> heart.test=read.csv("heart.test.ass3.2019.csv")
> prob = predict(step.fit.bic, heart.test, type = "response")
Warning message:
In predict.lm(object, newdata, se.fit, scale = 1, type = if (type ==  :
  prediction from a rank-deficient fit may be misleading
> prob[45]
       45
0.7102424
> odd= prob[45]/(1-prob[45])
> odd
     45
2.45116
```

By applying the formula odd $= \frac{p}{1-p}$:

The odds of having heart-disease by using the stepwise logistic regression model is:

$\frac{0.7102424}{(1-0.7102424)}$ = 2.45116

Answer: Therefore, the odds of having heart disease for the patient in the 45th row of the dataset from the tree model that used cross-validation is 0.42857143 and the odds of having disease for the patient in the 45th row of the dataset from the stepwise logistic regression model is 2.45116. The predicted odds for the two models have a huge difference as the odds of having heart-disease is 0.42857143 for the tree model and the odds is 2.46116 for stepwise logistic regression model. The tree model has a lower odds which is 0.42857143 compared to the stepwise logistic regression model which has 2.45116.

1.9 Use the bootstrap procedure (use at least 5, 000 bootstrap replications) to find a confidence interval for the probability of having heart disease for patient in the 45th row in the test data. Discuss this confidence interval in comparison to the predicted probabilities of having heart disease for both the logistic regression model and the tree model.

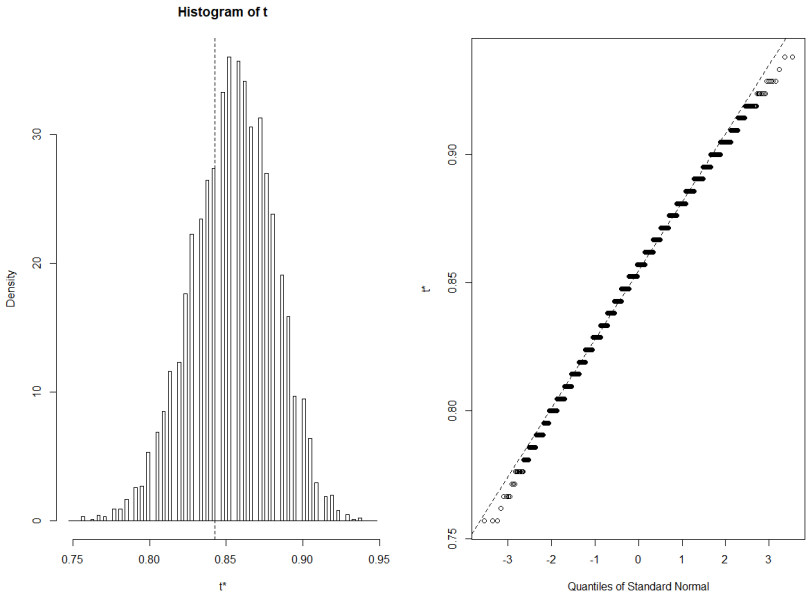
```
> #1.9
> boot.prob = function(formula, data, indices)
+ {
+   # Create a bootstrapped version of our data
+   d = data[indices,]
+
+   # Fit a logistic regression to the bootstrapped data
+   fit = glm(formula, data=d, family=binomial)
+
+   # Compute the AUC and return it
+   dataset=heart.test[45,]
+   rv = predict(fit,dataset,type="response")
+   return(rv)
+ }
> bs = boot(data=heart.train, statistic=boot.prob, R=5000, formula=HD ~ CP + EXANG  +OLDPEAK + CA + THAL)
There were 15 warnings (use warnings() to see them)
> boot.ci(bs,conf=0.95,type="bca")
BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 5000 bootstrap replicates

CALL :
boot.ci(boot.out = bs, conf = 0.95, type = "bca")

Intervals :
Level       BCa
95%   ( 0.3790,  0.8922 )
Calculations and Intervals on Original Scale
```

Answer: The confidence interval for the probability of having heart-disease for patient in the 45th row in the test data is (0.3790, 0.8922). It is 95% confident that the probability of having disease for the patient in the 45th row in the test data is fall between the range 0.3790 and 0.8922. The predicted probabilities of having heart-disease for the stepwise logistic regression model is 0.7102424 and for the tree model is 0.30. Therefore, we can say that the predicted probabilities of heart-disease for the stepwise logistic regression model is more accurate compared to the predicted probabilities of heart-disease for the tree model because the predicted probabilities of heart-disease for the stepwise logistic regression model falls within the confidence interval for the probability of having heart-disease for patient in the 45th row in the test data whereas the predicted probabilities for the tree model falls outside the confidence interval.

1.10 Use the bootstrap to compute a 95% confidence interval for the classification accuracy of the logistic regression model using the predictors selected by BIC. Plot the bootstrap results.

```
> #1.10
> boot.ca = function(formula, data, indices)
+ {
+   # Create a bootstrapped version of our data
+   d = data[indices,]
+
+   # Fit a logistic regression to the bootstrapped data
+   fit = glm(formula, d, family=binomial)
+
+   # Compute the AUC and return it
+   target = as.character(fit$terms[[2]])
+   rv = my.pred.stats(predict(fit,d,type="response"), d[,target], display=F)
+   return(rv$ca)
+ }
> bs = boot(data=heart.train, statistic=boot.ca, R=5000, formula=HD ~ CP + EXANG  +OLDPEAK + CA + THAL)
There were 12 warnings (use warnings() to see them)
> boot.ci(bs,conf=0.95,type="bca")
BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 5000 bootstrap replicates

CALL :
boot.ci(boot.out = bs, conf = 0.95, type = "bca")

Intervals :
Level       BCa
95%   ( 0.7587,  0.8762 )
Calculations and Intervals on Original Scale
Some BCa intervals may be unstable
> plot(bs)
```

The histogram of the bootstrapped classification accuracies is shown below:




```
> #1.7
> my.pred.stats(predict(step.fit.bic,heart.train,type="response"), heart.train$HD)
------------------------------------------------------------------------------
Performance statistics:

Confusion matrix:

    target
pred   N   Y
   N 101  20
   Y  13  76

Classification accuracy = 0.8428571
Sensitivity             = 0.7916667
Specificity             = 0.8859649
Area-under-curve        = 0.9063414
Logarithmic loss        = 78.31014
```

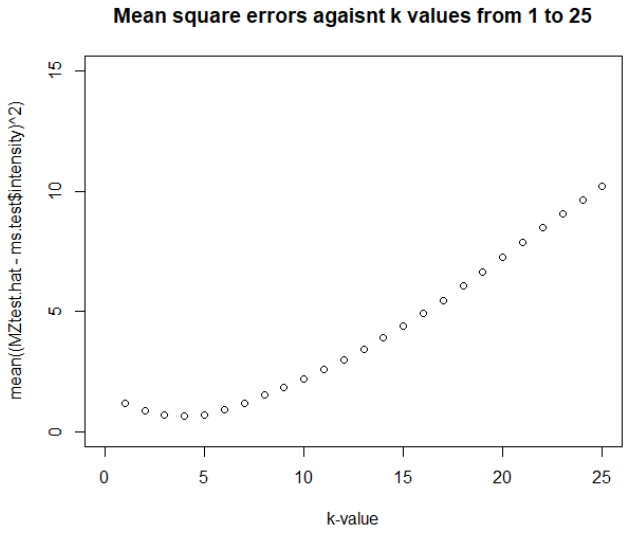
Answer: As it can be seen from the graph, the confidence interval for the classification accuracy of the logistic regression model using the predictors selected by BIC falls between the range 0.7587 and 0.8762. The classification accuracy for the stepwise logistic regression model is 0.8428571 which falls within the confidence interval. It is 95% confident that the classification accuracy is accurate as the classification accuracy value falls within the range of 0.7857 and 0.8762.

Task 2

2.1a Use k-NN to estimate the values of the spectrum and compute the mean squared error between your estimates of the spectrum and the true values in ms.test$intensity. Plot these errors against the various values of k from 1 to 25.

```
> #2.1.a
> ms.train = read.csv("ms.train.ass3.2019.csv", header=T)
> ms.test = read.csv("ms.test.ass3.2019.csv", header=T)
> mZtest.hat = fitted( kknn(intensity~ ., ms.train, ms.test, kernel = 'optimal', k = 1) )
> plot(1,mean((MZtest.hat - ms.test$intensity)^2),xlab="k-value",xlim = c(0,25),ylim = c(0,15), main="Mean square errors agaisnt k values from 1 to 25")
> error = mean((mZtest.hat - ms.test$intensity)^2)
> cat("when k-value = 1,","Mean Squared Error:",error,"\n")
when k-value = 1, Mean Squared Error: 1.316759
> for (i in 2:25) {
+   mZtest.hat = fitted( kknn(intensity~ ., ms.train, ms.test, kernel = 'optimal', k = i) )
+   error = mean((mZtest.hat - ms.test$intensity)^2)
+   cat("when k-value =",i,"Mean Squared Error:",error,"\n")
+   points(i,error)
+ }
when k-value = 2 Mean Squared Error: 0.8557929
when k-value = 3 Mean Squared Error: 0.6945513
when k-value = 4 Mean Squared Error: 0.6452697
when k-value = 5 Mean Squared Error: 0.717139
when k-value = 6 Mean Squared Error: 0.9091328
when k-value = 7 Mean Squared Error: 1.201222
when k-value = 8 Mean Squared Error: 1.524736
when k-value = 9 Mean Squared Error: 1.853013
when k-value = 10 Mean Squared Error: 2.20856
when k-value = 11 Mean Squared Error: 2.594765
when k-value = 12 Mean Squared Error: 3.002731
when k-value = 13 Mean Squared Error: 3.434071
when k-value = 14 Mean Squared Error: 3.892916
when k-value = 15 Mean Squared Error: 4.386852
when k-value = 16 Mean Squared Error: 4.915616
when k-value = 17 Mean Squared Error: 5.469157
when k-value = 18 Mean Squared Error: 6.049199
when k-value = 19 Mean Squared Error: 6.64908
when k-value = 20 Mean Squared Error: 7.258814
when k-value = 21 Mean Squared Error: 7.866367
when k-value = 22 Mean Squared Error: 8.467786
when k-value = 23 Mean Squared Error: 9.060099
when k-value = 24 Mean Squared Error: 9.638365
when k-value = 25 Mean Squared Error: 10.19935
```

The values above are the mean squared error from k-value 1 to k-value 25.



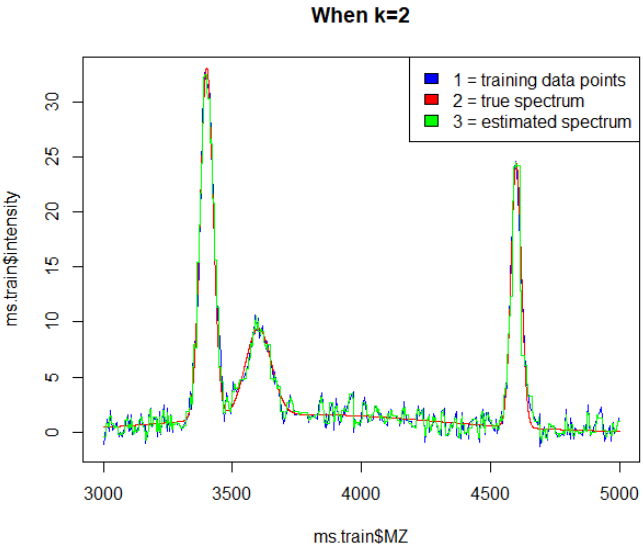
The plot above shows the mean square errors against k-value from 1 to 25.

2.1b Produce 4 graphs showing the training data points, the true spectrum and the estimated spectrum produced by k-NN method for 4 different values of k, for k=2, k=5, k=10 and k=25.

The following R code below shows that the graph produced by k-NN method for k-value = 2:

```
#when k=2
mZtest1.hat = fitted( kknn(intensity~ ., ms.train, ms.test, kernel = 'optimal', k = 2))
error1 = mean((mZtest1.hat - ms.test$intensity)^2)

plot(ms.train$MZ, ms.train$intensity,type='l',col="blue", main="when k=2")
lines(ms.test$MZ, ms.test$intensity,type='l',col="red")
lines(ms.test$MZ, mZtest1.hat,type='l',col="green")
legend("topright",c("1 = training data points","2 = true spectrum","3 = estimated spectrum"),fill=c("blue","red","green"))
```

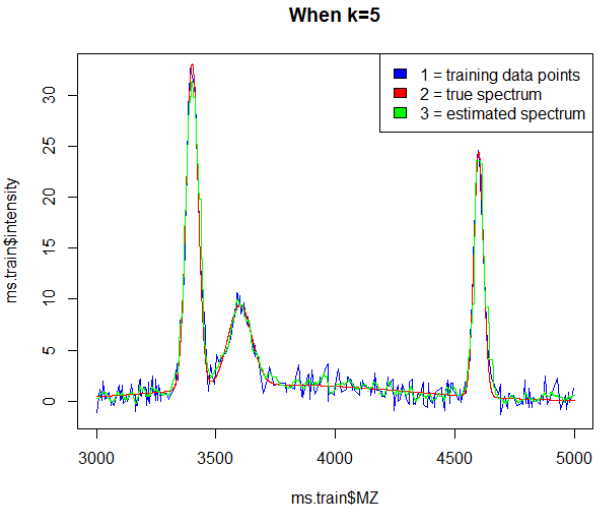
The following graph is produced by k-NN method for k-value = 2:



The following R code below shows that the graph produced by k-NN method for k-value = 5:

```
> #when k=5
> mZtest2.hat = fitted( kknn(intensity~ ., ms.train, ms.test, kernel = 'optimal', k = 5))
> error2 = mean((mZtest2.hat - ms.test$intensity)^2)
> plot(ms.train$MZ, ms.train$intensity,type='l',col="blue", main="When k=5")
> lines(ms.test$MZ, ms.test$intensity,type='l',col="red")
> lines(ms.test$MZ, mZtest2.hat,type='l',col="green")
> legend("topright",c("1 = training data points","2 = true spectrum","3 = estimated spectrum"),fill=c("blue","red","green"))
```

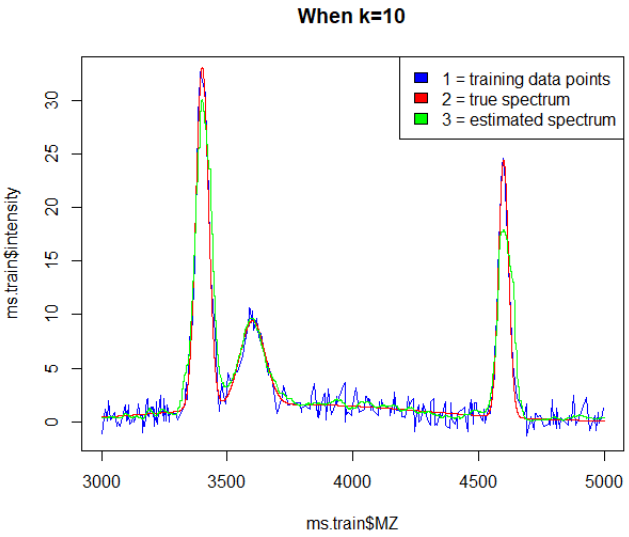
The following graph is produced by k-NN method for k-value = 5:



The following R code below shows that the graph produced by k-NN method for k-value = 10:

```
> #when k=10
> mZtest3.hat = fitted( kknn(intensity~ ., ms.train, ms.test, kernel = 'optimal', k = 10))
> error3 = mean((mZtest3.hat - ms.test$intensity)^2)
> plot(ms.train$MZ, ms.train$intensity,type='l',col="blue", main="When k=10")
> lines(ms.test$MZ, ms.test$intensity,type='l',col="red")
> lines(ms.test$MZ, mZtest3.hat,type='l',col="green")
> legend("topright",c("1 = training data points","2 = true spectrum","3 = estimated spectrum"),fill=c("blue","red","green"))
```

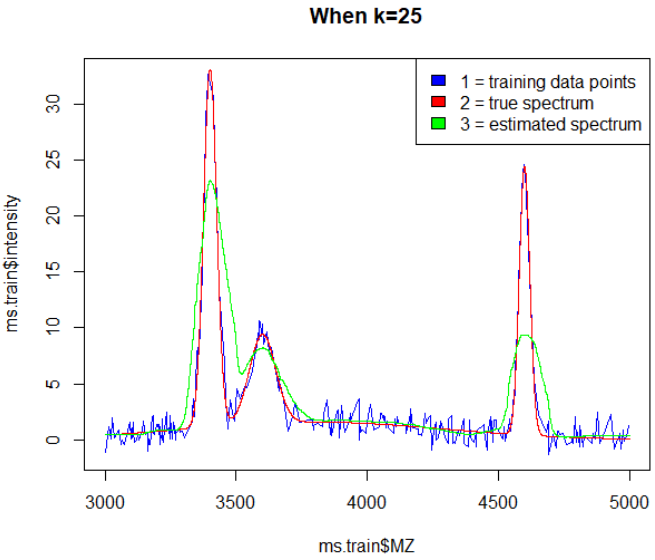
The following graph is produced by k-NN method for k-value = 10:



The following R code below shows that the graph produced by k-NN method for k-value = 25:

```
> #when k=25
> mZtest4.hat = fitted( kknn(intensity~ ., ms.train, ms.test, kernel = 'optimal', k = 25))
> error4 = mean((mZtest4.hat - ms.test$intensity)^2)
> plot(ms.train$MZ, ms.train$intensity,type='l',col="blue", main="When k=25")
> lines(ms.test$MZ, ms.test$intensity,type='l',col="red")
> lines(ms.test$MZ, mZtest4.hat,type='l',col="green")
> legend("topright",c("1 = training data points","2 = true spectrum","3 = estimated spectrum"),fill=c("blue","red","green"))
```

The following graph is produced by k-NN method for k-value = 25:



2.1c Discuss, qualitatively, and quantitatively (in terms of mean-squared error on the true spectrum) the four different estimates of the spectrum.

When k=2, the mean squared error is 0.8557929

When k=5, the mean squared error is 0.717139

When k=10, the mean squared error is 2.20856

When k=25, the mean squared error is 10.19935

Answer: As you can see from the values above, when k=5 the mean squared error is the lowest compared to the other mean squared errors when k=2, k=10 and k=25. The estimated spectrum curve is getting lower and lower which means it is getting less accurate because it's getting further away from the true spectrum.

2.2 Use the cross-validation functionality in the kknn package to select an estimate of the best value of k. What value of k does the method select?

```
> #2.2
> knn = train.kknn(intensity ~ ., data = ms.train, kmax=25, kernel="optimal")
> mstest.cross = fitted( kknn(intensity ~ ., ms.train, ms.test, kernel = knn$best.parameters$kernel, k = knn$best.parameters$k) )
> cat("k-value",knn$best.parameters$k,"is selected")
k-value 3 is selected
```

```
when k-value = 1, Mean Squared Error: 10.19935

when k-value = 2 Mean Squared Error: 0.8557929
when k-value = 3 Mean Squared Error: 0.6945513
when k-value = 4 Mean Squared Error: 0.6452697
when k-value = 5 Mean Squared Error: 0.717139
when k-value = 6 Mean Squared Error: 0.9091328
when k-value = 7 Mean Squared Error: 1.201222
when k-value = 8 Mean Squared Error: 1.524736
when k-value = 9 Mean Squared Error: 1.853013
when k-value = 10 Mean Squared Error: 2.20856
when k-value = 11 Mean Squared Error: 2.594765
when k-value = 12 Mean Squared Error: 3.002731
when k-value = 13 Mean Squared Error: 3.434071
when k-value = 14 Mean Squared Error: 3.892916
when k-value = 15 Mean Squared Error: 4.386852
when k-value = 16 Mean Squared Error: 4.915616
when k-value = 17 Mean Squared Error: 5.469157
when k-value = 18 Mean Squared Error: 6.049199
when k-value = 19 Mean Squared Error: 6.64908
when k-value = 20 Mean Squared Error: 7.258814
when k-value = 21 Mean Squared Error: 7.866367
when k-value = 22 Mean Squared Error: 8.467786
when k-value = 23 Mean Squared Error: 9.060099
when k-value = 24 Mean Squared Error: 9.638365
when k-value = 25 Mean Squared Error: 10.19935
```

Answer: The k-value 3 is selected by using the cross-validation functionality in the kknn package. The lowest mean squared error is found when k=4 by using the tree model from k-value=1 to k-value=25 which is not the same as by using the cross-validation functionality method. This means that it does not need to go through from k=1 to k=25 to find the optimum mean squared error but instead it can use the cross-validation method to find the optimum lowest mean squared error for the k max value provided.

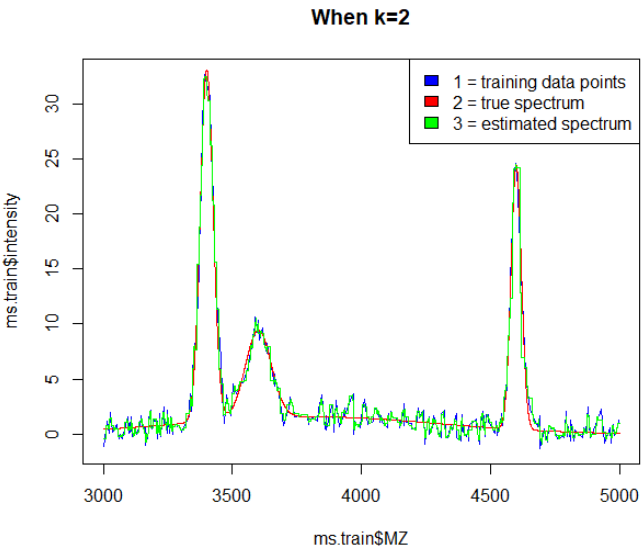
2.3 Using the estimation of the curve produced in the previous question and provide an estimation of the variance of the sensor/measurement noise that has corrupted our intensity measurements.

```
> #2.3
> mZtest.hat = fitted( kknn(intensity~ ., ms.train, ms.test, kernel = 'optimal', k = 3))
> var(mZtest.hat)
[1] 32.06271
```

Answer: By using the estimation of the curve produced, the variance of the sensor is 32.06271.

2.4 Do any of the estimated spectra achieve our aim of providing a smooth, low noise estimate of background level as well as accurate estimation of the peaks? Explain why the k-NN method can achieve, or not achieve, this aim.

Answer: Yes, as it can be seen below, the graph when k=2 achieve our aim of providing a smooth, low noise estimate of background level as well as accurate estimation of the peaks. This can be seen from the graph that the estimated spectrum curve and the true spectrum curve are almost the same as the estimated spectrum reaches the peak of the true spectrum. The reason why k-NN method can achieve this aim is because first it calculates the distance then move on to find the closest neighbors and vote for labels.



2.5 Produce a plot, as above, with the predicted intensity values for the MZ values in ms.test.csv using the final tree found by CV. Compare this qualitatively, and quantitatively (in terms of mean-squared error on the true spectrum) to the k-NN method.

R -code below shows how to get the plot with the predicted intensity values and the mean squared errors on the true spectrum:

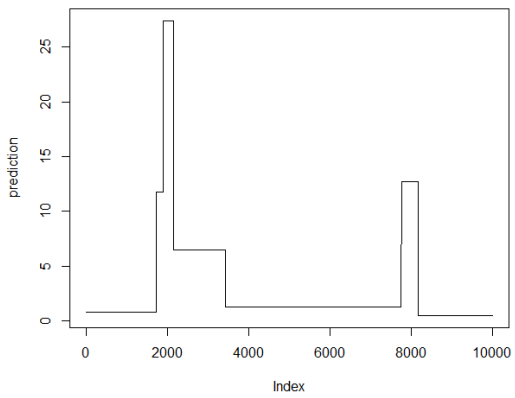
By using the tree model, the mean squared error is 4.723485.

```
> #2.5
> ms.train = read.csv("ms.train.ass3.2019.csv", header=T)
> ms.test = read.csv("ms.test.ass3.2019.csv", header=T)
> cv = learn.tree.cv(intensity~.,data=ms.train,nfolds=10,m=1000)
> ms.train1=rpart(intensity ~ .,ms.train)
> prune.rpart(tree=ms.train1, cp = cv$best.cp)
n= 300

node), split, n, deviance, yval
      * denotes terminal node

 1) root 300 12468.64000  3.4964650
   2) MZ>=3685.98 176  2583.11800  1.8598020
     4) MZ< 4549.96 122   111.36300  1.2593890 *
     5) MZ>=4549.96 54  2328.41100  3.2162910
      10) MZ>=4633.6 42    55.72858  0.4985788 *
      11) MZ< 4633.6 12   876.73650 12.7282800 *
   3) MZ< 3685.98 124  8744.92800  5.8194720
     6) MZ< 3346.1 69    51.86435  0.7995215 *
     7) MZ>=3346.1 55  4772.87600 12.1172300
      14) MZ>=3430.16 35   288.52990  6.5061230 *
      15) MZ< 3430.16 20  1453.96400 21.9366600
        30) MZ< 3376.53 7   158.97760 11.7976000 *
        31) MZ>=3376.53 13   187.90340 27.3961500 *
> prediction = predict(cv$best.tree,ms.test)
> plot(prediction,type="l")
> mean((predict(ms.train1, ms.test) - ms.test$intensity)^2)
[1] 4.723485
```

By using the KNN method, the mean squared error is 0.6945513.

```
> mZtest.hat = fitted( kknn(intensity~ ., ms.train, ms.test, kernel = 'optimal', k = 3) )
> error = mean((mZtest.hat - ms.test$intensity)^2)
> cat("when k-value = 3,","Mean Squared Error:",error,"\n")
when k-value = 3, Mean Squared Error: 0.6945513
```

The graph below is plotted with the predicted intensity values for the MZ values in ms.test.csv using the final tree found by CV:



Answer: By using the KNN method, the mean squared error is 0.6945513 whereas by using the tree model method, the mean squared error is 4.723485. So, the KNN method is better than the tree model method as it has a lower mean squared error compared to the tree model method.

2.6 Compare and contrast the decision tree approach with the k-NN approach on this one-dimensional problem. Thinking about the way in which both methods work, discuss how they are similar and they are different.

Answer: The difference between KNN and Decision tree is that KNN is unsupervised and Decision tree is supervised. Both KNN and Decision tree are used for classification but KNN can be used for clustering too. KNN determines neighbourhoods, so there must be a distance metric which means it must be numeric. Since KNN require distance metric and distance metric may be affected by varying scales between attributes and high-dimensional space. This means that KNN works well with small number of input variables but as the numbers of variables grow K-NN algorithm struggles to predict the output of new data point. On the other hand, Decision tree predicts a class for a given input vector and the attributes can be numerical or nominal. Therefore, KNN is used if you want to find similar examples and Decision tree is used if you want to classify examples.

2.7 Using the outputs of the bootstrap function boot(), plot an approximate 95% confidence interval for the estimated spectrum along with the true spectrum and the average of the bootstrap estimates of the spectra. Discuss the confidence interval in comparison to the true spectrum.

```
> #2.7
> bs.mean=boot(ms.train$intensity,function(x,I) { return(mean(x[I])) }, 100)
> bs.mean

ORDINARY NONPARAMETRIC BOOTSTRAP


Call:
boot(data = ms.train$intensity, statistic = function(x, I) {
    return(mean(x[I]))
}, R = 100)


Bootstrap Statistics :
    original      bias    std. error
t1* 3.496465 -0.02754526   0.3905884
> boot.ci(bs.mean, 0.95, type="basic")
BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 100 bootstrap replicates

CALL :
boot.ci(boot.out = bs.mean, conf = 0.95, type = "basic")

Intervals :
Level      Basic
95%   ( 2.550,  4.135 )
Calculations and Intervals on Original Scale
Some basic intervals may be unstable
> plot(bs.mean)
```

Answer: The confidence interval for the estimated spectrum for the MZ values in ms.test along with the true spectrum in ms.test and the average of the bootstrap estimates of the spectra falls between the range 2.550 and 4.135. The predicted value for the true spectrum is 2.893 and therefore we are 95% confident that the predicted value for the estimated spectrum for the MZ values in ms.test along with the true spectrum in ms.test and the average of the bootstrap estimates of the spectra is accurate because it falls within the confidence interval.