Student Name: Foo Kai Yan
Student ID: 33085625
Student Email: kfoo0012@student.monash.edu

FIT2086 Modelling for data analysis

Assignment 3

Question 1

Question 1.1

```
> ##################################################################
> # Assignment 3 Question 1.1
> ##################################################################
> # Load the given data
> housing_data <- read.csv("housing.2023.csv")
> # summary() used to examine the data set
> summary(housing_data)
      crim                zn              indus            chas            nox
 Min.   : 0.00632   Min.   :  0.000   Min.   : 1.32   Min.   :0.000   Min.   :0.3890
 1st Qu.: 0.08889   1st Qu.:  0.000   1st Qu.: 5.19   1st Qu.:0.000   1st Qu.:0.4610
 Median : 0.30940   Median :  0.000   Median : 9.90   Median :0.000   Median :0.5380
 Mean   : 3.90795   Mean   :  9.214   Mean   :11.45   Mean   :0.072   Mean   :0.5570
 3rd Qu.: 3.65650   3rd Qu.:  0.000   3rd Qu.:18.10   3rd Qu.:0.000   3rd Qu.:0.6292
 Max.   :88.97600   Max.   :100.000   Max.   :27.74   Max.   :1.000   Max.   :0.8710
       rm              age              dis              rad              tax
 Min.   :4.138   Min.   :  6.20   Min.   : 1.130   Min.   : 1.000   Min.   :187.0
 1st Qu.:5.888   1st Qu.: 46.83   1st Qu.: 2.103   1st Qu.: 4.000   1st Qu.:277.0
 Median :6.170   Median : 79.85   Median : 3.101   Median : 5.000   Median :329.0
 Mean   :6.295   Mean   : 70.21   Mean   : 3.635   Mean   : 9.584   Mean   :404.5
 3rd Qu.:6.630   3rd Qu.: 94.60   3rd Qu.: 4.765   3rd Qu.:24.000   3rd Qu.:666.0
 Max.   :8.780   Max.   :100.00   Max.   :12.127   Max.   :24.000   Max.   :711.0
    ptratio          lstat            medv
 Min.   :12.60   Min.   : 1.730   Min.   : 6.30
 1st Qu.:17.00   1st Qu.: 7.545   1st Qu.:17.10
 Median :18.70   Median :11.330   Median :21.40
 Mean   :18.35   Mean   :12.899   Mean   :22.81
 3rd Qu.:20.20   3rd Qu.:17.277   3rd Qu.:25.00
 Max.   :21.20   Max.   :37.970   Max.   :50.00
> # Fit a multiple linear model
> housing_multiple_linear_model <- lm(medv ~ ., data = housing_data)
> # Summary of the linear model
> summary(housing_multiple_linear_model)
```

```
Call:
lm(formula = medv ~ ., data = housing_data)

Residuals:
     Min       1Q   Median       3Q      Max
-17.9480  -2.7966  -0.5589   1.5896  26.2270

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  34.054337   7.568558   4.499 1.07e-05 ***
crim         -0.115818   0.041915  -2.763 0.006174 **
zn            0.018561   0.021190   0.876 0.381961
indus        -0.011274   0.087587  -0.129 0.897691
chas          4.163521   1.299647   3.204 0.001544 **
nox         -16.722652   6.154586  -2.717 0.007071 **
rm            4.501521   0.688705   6.536 3.83e-10 ***
age           0.001457   0.020603   0.071 0.943690
dis          -1.163294   0.315727  -3.684 0.000284 ***
rad           0.291680   0.112473   2.593 0.010096 *
tax          -0.012387   0.006284  -1.971 0.049871 *
ptratio      -0.960017   0.199722  -4.807 2.73e-06 ***
lstat        -0.480698   0.079723  -6.030 6.26e-09 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 5.164 on 237 degrees of freedom
Multiple R-squared:  0.7089,    Adjusted R-squared:  0.6942
F-statistic:  48.1 on 12 and 237 DF,  p-value: < 2.2e-16

> # Extract coefficients and p-values
> coefficients_summary <- summary(housing_multiple_linear_model)$coefficients
> # Find significant predictors (p-value < 0.05)
> significant_predictors <- coefficients_summary[coefficients_summary[, "Pr(>|t|)"] < 0.05, ]
> # Sort predictors by absolute coefficient magnitude (significant)
> strongest_predictors <- significant_predictors[order(abs(significant_predictors[, "Pr(>|t|)"]), decreasing = FALSE), ]
> # Display the significant predictors (those with *)
> significant_predictors
               Estimate  Std. Error   t value     Pr(>|t|)
(Intercept)  34.0543367 7.568557561  4.499449 1.067882e-05
crim         -0.1158180 0.041915426 -2.763135 6.174281e-03
chas          4.1635213 1.299647482  3.203577 1.543615e-03
nox         -16.7226516 6.154586130 -2.717104 7.071490e-03
rm            4.5015206 0.688704777  6.536212 3.829556e-10
dis          -1.1632937 0.315727259 -3.684489 2.839809e-04
rad           0.2916797 0.112472683  2.593338 1.009643e-02
tax          -0.0123874 0.006284366 -1.971146 4.987072e-02
ptratio      -0.9600173 0.199721995 -4.806768 2.725225e-06
lstat        -0.4806976 0.079723468 -6.029562 6.258366e-09
> # Display the three highest/strongest predictors (those with ***)
> strongest_predictors[1:3, ]
          Estimate Std. Error   t value     Pr(>|t|)
rm       4.5015206 0.68870478  6.536212 3.829556e-10
lstat   -0.4806976 0.07972347 -6.029562 6.258366e-09
ptratio -0.9600173 0.19972200 -4.806768 2.725225e-06
>
```

Average number of rooms per dwelling (rm), Percentage of “lower status” of the population (lstat) and Pupil-teacher ratio (ptratio) are the three variables appear to be the strongest predictors of housing price. From the summary output of housing_multiple_linear_model, we can see that there are 5 variables that have been denoted with 3 asterick (*) which shows that its is a relatively significant predictor as predictors with low p-values are most likely to be strongly associated with the median house value. The column “Pr(>|t|)” shows the p-values of the variables. The variables were

sorted by their magnitude in an ascending order from smallest to largest p-value in strongest_predictors so the first 3 of the lists of strongest_predictors are the variables that is the strongest predictors of housing price.

Question 1.2

```
> ################################################################
> # Assignment 3 Question 1.2
> ################################################################
> # Define the significant level (α) and the number of predictors (p)
> alpha <- 0.05
> num_predictors <- length(coefficients_summary[, "Pr(>|t|)"])
> # Calculate the Bonferroni-corrected significance level
> alpha_bonferroni <- alpha / num_predictors
> # Identify significant predictors using Bonferroni correction
> significant_predictors_bonferroni <- coefficients_summary[coefficients_summary[, "Pr(>|t|)"] <= alpha_bonferroni, ]
> # Display the significant predictors with Bonferroni correction
> significant_predictors_bonferroni
              Estimate Std. Error   t value     Pr(>|t|)
(Intercept) 34.0543367 7.56855756  4.499449 1.067882e-05
chas         4.1635213 1.29964748  3.203577 1.543615e-03
rm           4.5015206 0.68870478  6.536212 3.829556e-10
dis         -1.1632937 0.31572726 -3.684489 2.839809e-04
ptratio     -0.9600173 0.19972200 -4.806768 2.725225e-06
lstat       -0.4806976 0.07972347 -6.029562 6.258366e-09
>
```

The equation used here is the significant level (α) / number of predictors (p). Using the Bonferroni procedure allow the reduction of risk of having false positives whereby a smaller p-value is required for a variable to be declare as a significant predictor. It is observed that the variables with 2 or more * from the summary output of housing_multiple_linear_model is displayed here as output as these variables have a lower p-value compared to the rest of the variables. Hence, chas, rm, dist, ptratio and lstat are the predictors which are associated change when Bonferroni procedure with α = 0.05.

Question 1.3

```
> ################################################################
> # Assignment 3 Question 1.3
> ################################################################
> # Find the coefficient for 'crim'
> coefficient_crim <- coefficients_summary["crim", "Estimate"]
> coefficient_crim
[1] -0.115818
> # Find the coefficient for 'chas'
> coefficient_chas <- coefficients_summary["chas", "Estimate"]
> coefficient_chas
[1] 4.163521
>
```

Per-capita crime rate (crim) is -0.115818 which is less than 0 which shows that an increase in the per-capita crime rate is associated with a decrease in median house price.

Suburb front the Charles River (chas) is 4.163521 which is more than 0 which shows that having frontage on the Charles River is associated with an increase in median house price.

Question 1.4

```
> ################################################################
> # Assignment 3 Question 1.4
> ################################################################
> # Perform stepwise selection with BIC
> stepwise_selection_model <- step(housing_multiple_linear_model, k = log(nrow(housing_data)), direction = "both")

> stepwise_selection_model
```

```
Call:
lm(formula = medv ~ chas + nox + rm + dis + ptratio + lstat,
    data = housing_data)

Coefficients:
(Intercept)         chas          nox           rm          dis      ptratio        lstat
    29.1927       4.5991     -17.3765       4.8206      -0.9359      -0.9591      -0.4947
```

```
> # Display the final model summary
> summary(stepwise_selection_model)

Call:
lm(formula = medv ~ chas + nox + rm + dis + ptratio + lstat,
    data = housing_data)

Residuals:
     Min       1Q   Median       3Q      Max
-17.9664  -2.9608  -0.6105   1.8037  26.9903

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  29.19267    6.67115   4.376 1.80e-05 ***
chas          4.59911    1.30302   3.530 0.000498 ***
nox         -17.37651    5.06186  -3.433 0.000702 ***
rm            4.82065    0.64361   7.490 1.27e-12 ***
dis          -0.93594    0.27030  -3.463 0.000632 ***
ptratio      -0.95914    0.16483  -5.819 1.86e-08 ***
lstat        -0.49472    0.07408  -6.678 1.63e-10 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 5.239 on 243 degrees of freedom
Multiple R-squared:  0.6928,    Adjusted R-squared:  0.6853
F-statistic: 91.35 on 6 and 243 DF,  p-value: < 2.2e-16

>
```

After pruning out potentially unimportant variables with using stepwise selection procedure with the BIC criterionl the final regression equation obtained is $E[medv] = 29.19267 + 4.5991 * chas - 17.37651 * nox + 4.82065 * rm - 0.93594 * dis - 0.95914 * ptratio$ which is extrated from stepwise_seelction_model summary.

Question 1.5

```
> ##################################################################
> # Assignment 3 Question 1.5
> ##################################################################
> # Riverfront Development (chas)
> # Air Quality (nox)
> # School Quality (ptratio)
> # Income Levels (lstat)
> # Room Count (rm)
> # Proximity to Employment (dis)
>
```

The model that found in Question 1.4 suggest that the council should promote riverfront development like the suburb front the Charles River as the coefficient of doing so like the suburb front the Charles River is 4.59911 which implies that

having riverfront development is associated with higher median house values as it may have increased the property values due to scenary of Charles River.

The model that found in Question 1.4 also suggest that the council should control the air quality by reducing pollution as from the data it is seen that the coefficient of nitrogen oxide concentration is -17.37651 which implies that higher levels of nitrogen oxide are associated with lower median house values as it may have decreased the property values due to severe air quality.

The model also suggests the council to hire more teachers, so each teacher is responsible for teaching a small number of students which allows the teacher to focus and pay more attention to each students' learning. The coefficient of pupil-teacher ratio (ptratio) is -0.95914 which implies that there is a better school quality with a lower pupil-teacher ratio. If the council have schools that are better in the suburb, then families with children would want their children to attend the said schools and for convenience purposes, the families will want to live closer so the demands for house in the said suburb is higher and hence increasing the property values.

Other than that, the model also suggests that the lower percentage of lower-income residents is associated with higher house values due to the coefficient of lstat being -0.49471. So, if the council market the property to have an affordable and suitable pricing range for a wide range of individuals then these individuals might want to live in the suburb that is within their budget and hence the property values might increase as individuals would want to live there would offer a higher price to obtain the property in the said suburb.

The room count of the property from the model suggests that the council should have houses with more rooms as these houses with more rooms have a higher house value as shown in the model whereby the coefficient of rm is 4.82065. Hence, the council should have the properties to be constructed in a way that there is more room for each property.

Finally, the coefficient of dis which is the weighted distances to five Boston employment centres is -0.93594 which implies that houses closer to employment centers tend to have higher values which may be highly due to convenience. Hence, to increase the property values, councils should invest in having better public transportation or building more employment centres to allow residents easy access to workplace and more job opportunities.

Question 1.6

```
> ###################################################################
> # Assignment 3 Question 1.6
> ###################################################################
> new_suburb_data <- data.frame(
+   crim = 0.04741,
+   zn = 0,
+   indus = 11.93,
+   chas = 0,
+   nox = 0.573,
+   rm = 6.03,
+   age = 80.8,
+   dis = 2.505,
+   rad = 1,
+   tax = 273,
+   ptratio = 21,
+   lstat = 7.88
+ )
>
> # Predict the median house price for the new suburb using the stepwise selection model
> predicted_price <- predict(stepwise_selection_model, newdata = new_suburb_data)
>
> # Calculate the confidence interval
> confidence_interval <- predict(stepwise_selection_model, newdata = new_suburb_data, interval = "confidence")
>
> # Extract the lower and upper bounds of the confidence interval
> lower_bound <- confidence_interval[, "lwr"]
> lower_bound
[1] 20.30209
> upper_bound <- confidence_interval[, "upr"]
> upper_bound
[1] 23.53712
>
```

The confidence interval is 20.30209 – 23.53712 whereby the lower bound is 20.30209 and the upper bound is 23.53712.

Question 1.7

```
> ###################################################################
> # Assignment 3 Question 1.7
> ###################################################################
> # Fit a linear model with an interaction term
> interaction_model <- lm(medv ~ rm * dis, data = housing_data)
>
> # Summary of the interaction model
> summary(interaction_model)

Call:
lm(formula = medv ~ rm * dis, data = housing_data)

Residuals:
    Min      1Q  Median      3Q     Max
-20.897  -2.936   0.073   2.569  31.846

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -25.0515     7.2104  -3.474 0.000605 ***
rm            7.3103     1.1377   6.426 6.75e-10 ***
dis          -3.2006     2.0198  -1.585 0.114334
rm:dis        0.5837     0.3132   1.864 0.063580 .
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 6.548 on 246 degrees of freedom
Multiple R-squared:  0.5142,    Adjusted R-squared:  0.5083
F-statistic:  86.8 on 3 and 246 DF,  p-value: < 2.2e-16

>
```

rm = Average number of rooms per dwelling; dis = Weighted distances to five Boston employment centres

The coefficient of rm is 7.3103 while the coefficient of dis is -3.2006. The coefficient of the interaction between rm and dis is 0.5837. We can see that the p-value for the interaction between rm and dis is 0.063580 which is more than 0.05 which shows that there might be a weak interaction between dis and rm but its not significantly important to impact anything.

Question 2

Question 2.1

```
> ####################################################################
> # Assignment 3 Question 2.1
> ####################################################################
> # Classification problem; Presence of heart disease (HD)
> # Load the rpart package used in Studio 9
> library(rpart)
> library(pROC)
Type 'citation("pROC")' for a citation.

Attaching package: ‘pROC’

The following objects are masked from ‘package:stats’:

    cov, smooth, var

>
> # Load the given data
> heart_train_data <- read.csv("heart.train.2023.csv")
>
> # Load wrapper which is used to make some function usage easier
> source("wrappers.R")
>
> # summary() used to examine the data set
> summary(heart_train_data)
```

```
      AGE             SEX                CP              TRESTBPS          CHOL            FBS
 Min.   :29.0   Length:260         Length:260         Min.   :100.0   Min.   :126.0   Length:260
 1st Qu.:48.0   Class :character   Class :character   1st Qu.:120.0   1st Qu.:212.8   Class :character
 Median :56.0   Mode  :character   Mode  :character   Median :130.0   Median :244.0   Mode  :character
 Mean   :54.7                                         Mean   :132.8   Mean   :249.1
 3rd Qu.:61.0                                         3rd Qu.:140.0   3rd Qu.:276.2
 Max.   :76.0                                         Max.   :200.0   Max.   :564.0
   RESTECG             THALACH         EXANG              OLDPEAK         SLOPE                CA
 Length:260         Min.   : 71.0   Length:260         Min.   :0.000   Length:260         Min.   :0.0000
 Class :character   1st Qu.:133.8   Class :character   1st Qu.:0.000   Class :character   1st Qu.:0.0000
 Mode  :character   Median :152.0   Mode  :character   Median :0.800   Mode  :character   Median :0.0000
                    Mean   :149.4                      Mean   :1.065                      Mean   :0.6731
                    3rd Qu.:165.0                      3rd Qu.:1.600                      3rd Qu.:1.0000
                    Max.   :202.0                      Max.   :6.200                      Max.   :3.0000
    THAL               HD
 Length:260         Length:260
 Class :character   Class :character
 Mode  :character   Mode  :character
```

```
> # Load the required libraries
> library(rpart)
>
> # Fit a decision tree to the heart training data
> heart_train_data_decision_tree <- rpart(HD ~ ., data = heart_train_data)
> heart_train_data_decision_tree
n= 260

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

 1) root 260 125 N (0.51923077 0.48076923)
   2) THAL=Normal 140  34 N (0.75714286 0.24285714)
     4) CP=Atypical,NonAnginal,Typical 95  12 N (0.87368421 0.12631579) *
     5) CP=Asymptomatic 45  22 N (0.51111111 0.48888889)
      10) CA< 0.5 28   7 N (0.75000000 0.25000000)
        20) AGE< 58.5 18   1 N (0.94444444 0.05555556) *
        21) AGE>=58.5 10   4 Y (0.40000000 0.60000000) *
      11) CA>=0.5 17   2 Y (0.11764706 0.88235294) *
   3) THAL=Fixed.Defect,Reversible.Defect 120  29 Y (0.24166667 0.75833333)
     6) CA< 0.5 53  24 Y (0.45283019 0.54716981)
      12) EXANG=N 31  10 N (0.67741935 0.32258065)
        24) AGE>=51 20   3 N (0.85000000 0.15000000) *
        25) AGE< 51 11   4 Y (0.36363636 0.63636364) *
      13) EXANG=Y 22   3 Y (0.13636364 0.86363636) *
     7) CA>=0.5 67   5 Y (0.07462687 0.92537313) *
>
> # Visualize the decision tree
> plot(heart_train_data_decision_tree)
> text(heart_train_data_decision_tree, digits = 3)
>
> # Show the importance of each variable
> variable_importance_scores <- heart_train_data_decision_tree$variable.importance / max(heart_train_data_decision_tree$v
ariable.importance)
> variable_importance_scores
      THAL         CP    THALACH      EXANG         CA    OLDPEAK        AGE        SEX   TRESTBPS       CHOL
1.00000000 0.67776084 0.56160698 0.53488602 0.49288282 0.46811138 0.35657791 0.31629908 0.14729831 0.10097012
     SLOPE
0.03256035
>
> # Cross validation
> heart_train_data_cross_validation <- learn.tree.cv(HD ~ ., data = heart_train_data, nfolds = 10, m = 5000)
> heart_train_data_cross_validation
```

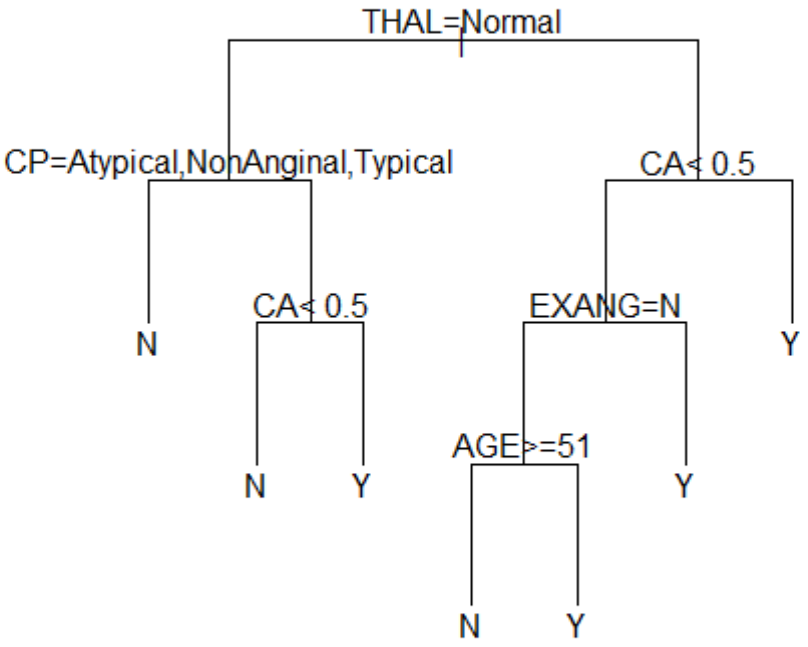
```
> heart_train_data_cross_validation
$best.tree
n= 260

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

 1) root 260 125 N (0.51923077 0.48076923)
   2) THAL=Normal 140  34 N (0.75714286 0.24285714)
     4) CP=Atypical,NonAnginal,Typical 95  12 N (0.87368421 0.12631579) *
     5) CP=Asymptomatic 45  22 N (0.51111111 0.48888889)
      10) CA< 0.5 28   7 N (0.75000000 0.25000000) *
      11) CA>=0.5 17   2 Y (0.11764706 0.88235294) *
   3) THAL=Fixed.Defect,Reversible.Defect 120  29 Y (0.24166667 0.75833333)
     6) CA< 0.5 53  24 Y (0.45283019 0.54716981)
      12) EXANG=N 31  10 N (0.67741935 0.32258065)
        24) AGE>=51 20   3 N (0.85000000 0.15000000) *
        25) AGE< 51 11   4 Y (0.36363636 0.63636364) *
      13) EXANG=Y 22   3 Y (0.13636364 0.86363636) *
     7) CA>=0.5 67   5 Y (0.07462687 0.92537313) *

$cv.stats
$cv.stats$cp
    1     2     3     4     5     6
0.496 0.052 0.044 0.024 0.016 0.010

$cv.stats$cv.err
        1         2         3         4         5         6
1.0197600 0.6102848 0.5264640 0.4439920 0.4277584 0.4325936

$cv.stats$n.leaves
[1] 1 2 4 6 7 8


$best.cp
    5
0.016

> plot.tree.cv(heart_train_data_cross_validation)
>
> # Get the best tree
> best_tree <- heart_train_data_cross_validation$best.tree
> best_tree
n= 260

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

 1) root 260 125 N (0.51923077 0.48076923)
   2) THAL=Normal 140  34 N (0.75714286 0.24285714)
     4) CP=Atypical,NonAnginal,Typical 95  12 N (0.87368421 0.12631579) *
     5) CP=Asymptomatic 45  22 N (0.51111111 0.48888889)
      10) CA< 0.5 28   7 N (0.75000000 0.25000000) *
      11) CA>=0.5 17   2 Y (0.11764706 0.88235294) *
   3) THAL=Fixed.Defect,Reversible.Defect 120  29 Y (0.24166667 0.75833333)
     6) CA< 0.5 53  24 Y (0.45283019 0.54716981)
      12) EXANG=N 31  10 N (0.67741935 0.32258065)
        24) AGE>=51 20   3 N (0.85000000 0.15000000) *
        25) AGE< 51 11   4 Y (0.36363636 0.63636364) *
      13) EXANG=Y 22   3 Y (0.13636364 0.86363636) *
     7) CA>=0.5 67   5 Y (0.07462687 0.92537313) *
>
```

The variables Age (AGE), Thallium scanning results (THAL), Chest pain type (CP), Number of major vessels colored by flourosopy (CA) and Exercise induced angina (EXANG) has been used in the best tree. “THAL” has been used to split “THAL = Normal” and “THAL = Fixed.Defect, Reversible.Defect”. “CP” has been used to split between “CP = Atypical, NonAnginal, Typical” and “CP = Asymptomatic”. “CA” has been used to split between “CA < 0.5” and “CA >= 0.5”. “EXANG” is used to split between “EXANG = N” and “EXANG = Y”.

The asterisk (*) denotes terminal nodes, which are also known as leaf nodes. The best tree have 7 terminal nodes (leaves).

Question 2.2

```
> ################################################################
> # Assignment 3 Question 2.2
> ################################################################
> # Plot the tree found by cross validation
> # Cross validation done in 2.1 is stored in the variable heart_train_data_cross_validation
> plot(best_tree, margin=0.2,uniform=T)
> text(heart_train_data_cross_validation$best.tree,pretty=12)
>
```



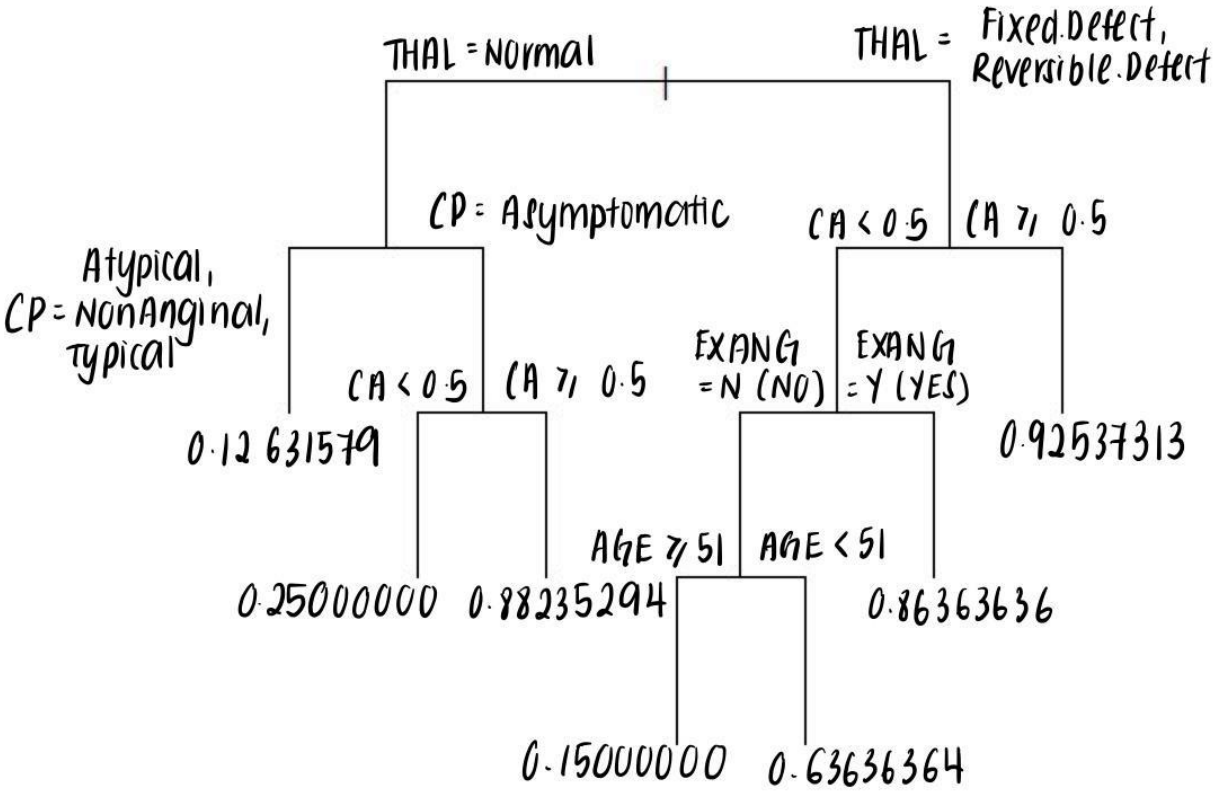
First and foremost, the starting point is at the highest position is the root node which is the entire dataset. The dataset is first split on the THAL variable on Normal with Fixed.Defect, Reversible.Defect.

If the data falls under the THAL = Normal category then that data is split to the left child would then be further split on the CP variable on whether the data falls under the CP = Asymptomatic or CP = Atypical, NonAnginal, Typical. If the data falls under the CP = Atypical, NonAnginal, Typical then that data does not have Heart Disease (HD) but if the data falls under the CP = Asymptomatic then it would be further split on whether CA is < 0.5 or >= 0.5. If data falls under the CA < 0.5 category then that data does not have Heart Disease (HD) but if CA >= 0.5 then that data does have Heart Disease (HD).

If the data falls under the THAL = Fixed.Defect, Reversible.Defect then that data is split to the right child would then be further split on the CA variable value whereby if CA >= 0.5 then that data does have Heart Disease (HD) but if CA < 0.5 then the data would be further split on its EXANG value. If the data EXANG value = Y which is Yes then that data does have Heart Disease (HD) but if the data EXANG value = N which is No then that data would still need to be further split whereby if the data age is >= 51 then that data does not have Heart Disease (HD) but if thet data age is < 51 then that data does have Heart Disease (HD).

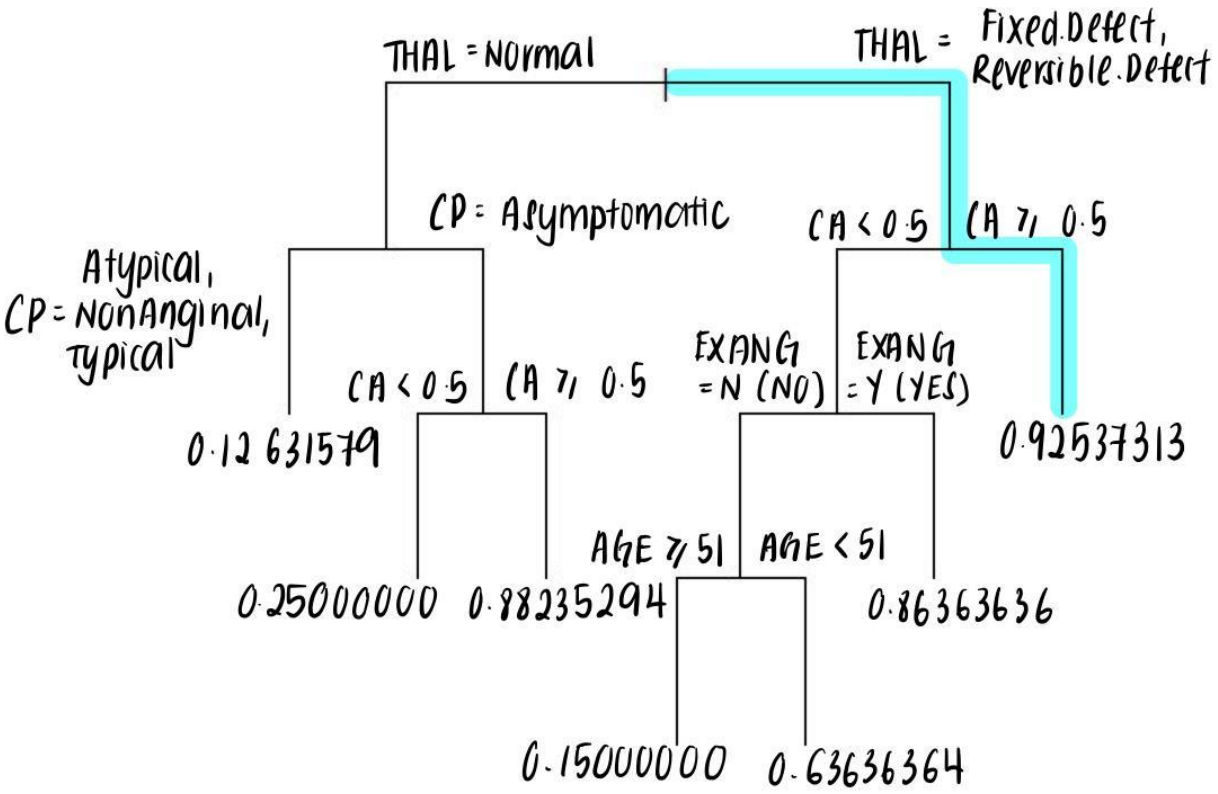
Question 2.3

```
> ##################################################################
> # Assignment 3 Question 2.3
> ##################################################################
> # Obtain the blank tree to be annotated
> plot(best_tree, margin=0.2,uniform=T)
>
```



Question 2.4

```
> ####################################################################
> # Assignment 3 Question 2.4
> ####################################################################
> best_tree
n= 260

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

 1) root 260 125 N (0.51923077 0.48076923)
   2) THAL=Normal 140  34 N (0.75714286 0.24285714)
     4) CP=Atypical,NonAnginal,Typical 95  12 N (0.87368421 0.12631579) *
     5) CP=Asymptomatic 45  22 N (0.51111111 0.48888889)
      10) CA< 0.5 28   7 N (0.75000000 0.25000000) *
      11) CA>=0.5 17   2 Y (0.11764706 0.88235294) *
   3) THAL=Fixed.Defect,Reversible.Defect 120  29 Y (0.24166667 0.75833333)
     6) CA< 0.5 53  24 Y (0.45283019 0.54716981)
      12) EXANG=N 31  10 N (0.67741935 0.32258065)
        24) AGE>=51 20   3 N (0.85000000 0.15000000) *
        25) AGE< 51 11   4 Y (0.36363636 0.63636364) *
      13) EXANG=Y 22   3 Y (0.13636364 0.86363636) *
     7) CA>=0.5 67   5 Y (0.07462687 0.92537313) *
```



According to the best_tree obtained, when the data have THAL = Fixed.Defect, Reversible.Defect and CA >= 0.5 the probability of having Heart Disease (HD) is the highest at 0.92537313.

Question 2.5

```
> ####################################################################
> # Assignment 3 Question 2.5
> ####################################################################
> # HD is not in binary data as shown with this code
> unique(heart_train_data$HD)
[1] "N" "Y"
>
> # Re-code "N" to 0 and "Y" to 1 in the 'HD' column
> # This is because we want to use family = binomial in glm
> heart_train_data$HD <- ifelse(heart_train_data$HD == "N", 0, 1)
>
> # Check if it is all modified to only 0 and 1
> unique(heart_train_data$HD)
[1] 0 1
>
> # Fit the logistic regression model
> heart_train_logistic_regression_model <- glm(HD ~ ., data = heart_train_data, family = binomial)
> heart_train_logistic_regression_model

Call:  glm(formula = HD ~ ., family = binomial, data = heart_train_data)

Coefficients:
          (Intercept)                   AGE                  SEXM            CPAtypical          CPNonAnginal
            -2.986357             -0.008904              1.468322             -0.989700             -1.740135
           CPTypical              TRESTBPS                  CHOL               FBS>120         RESTECGNormal
            -2.098109              0.024234              0.004740             -0.769233             -0.629625
     RESTECGST.T.Wave               THALACH                EXANGY               OLDPEAK               SLOPEFlat
            -0.186629             -0.023936              1.029208              0.394286              1.200481
             SLOPEUp                    CA            THALNormal  THALReversible.Defect
             0.414334              1.300509              0.177427              1.437050

Degrees of Freedom: 259 Total (i.e. Null);  241 Residual
Null Deviance:      360.1
Residual Deviance: 170.9        AIC: 208.9

> # Use stepwise selection with the BIC score to prune the logistic regression model
> stepwise_selection_model_BIC <- step(heart_train_logistic_regression_model, k = log(length(heart_train_data$HD)), direction
="both")

> stepwise_selection_model_BIC

Call:  glm(formula = HD ~ CP + THALACH + OLDPEAK + CA + THAL, family = binomial,
    data = heart_train_data)

Coefficients:
          (Intercept)            CPAtypical          CPNonAnginal              CPTypical               THALACH
              2.74052              -1.18588              -1.89032              -1.85305              -0.02349
              OLDPEAK                    CA            THALNormal  THALReversible.Defect
              0.57627               1.09854              -0.32528               1.45941

Degrees of Freedom: 259 Total (i.e. Null);  251 Residual
Null Deviance:      360.1
Residual Deviance: 194.1        AIC: 212.1
>
```

The final model that predict heart disease (HD) includes Chest Pain Type (CP), Maximum heart rate achieved (THALACH), Exercise induced ST depression relative to rest (OLDPEAK), Number of major vessels colored by flourosopy (CA) and Thallium scanning results (THAL).

Comparing to the tree estimated by cross validation, the tree uses more variables as predictors while the final model above chooses and pick the variables as predictors that are of higher importance. The most important predictor in the logistic regression model is CPNonAnginal with a coefficient of -1.89032 as shown in the output above.

Question 2.6

```
> ####################################################################
> # Assignment 3 Question 2.6
> ####################################################################
> stepwise_selection_model_BIC

Call:  glm(formula = HD ~ CP + THALACH + OLDPEAK + CA + THAL, family = binomial,
    data = heart_train_data)

Coefficients:
          (Intercept)             CPAtypical           CPNonAnginal               CPTypical                 THALACH
              2.74052               -1.18588               -1.89032               -1.85305                -0.02349
              OLDPEAK                     CA             THALNormal  THALReversible.Defect
              0.57627                1.09854               -0.32528                1.45941

Degrees of Freedom: 259 Total (i.e. Null);  251 Residual
Null Deviance:      360.1
Residual Deviance: 194.1        AIC: 212.1
>
```

Using the stepwise selection to find the logistic regression model, the final regression equation is $E[HD] = 2.74052 - 1.18588 * CPAtypical - 1.89032 * CPNonAnginal - 1.85305 * CPTypical - 0.02349 * THA$

Question 2.7

```
> ####################################################################
> # Assignment 3 Question 2.7
> ####################################################################
> # Load the given data
> heart_test_data <- read.csv("heart.test.2023.csv", stringsAsFactors = T)
>
> # Load my.prediction.stats.R to use my.pred.stats() function
> source("my.prediction.stats.R")
>
> # Compute the prediction statistics for both tree and step-wise logistic regression model on heart_test_data
> stepwise_selection_model_BIC

Call:  glm(formula = HD ~ CP + THALACH + OLDPEAK + CA + THAL, family = binomial,
    data = heart_train_data)

Coefficients:
          (Intercept)             CPAtypical           CPNonAnginal               CPTypical                 THALACH
              2.74052               -1.18588               -1.89032               -1.85305                -0.02349
              OLDPEAK                     CA             THALNormal  THALReversible.Defect
              0.57627                1.09854               -0.32528                1.45941

Degrees of Freedom: 259 Total (i.e. Null);  251 Residual
Null Deviance:      360.1
Residual Deviance: 194.1        AIC: 212.1
> best_tree
n= 260

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

 1) root 260 125 N (0.51923077 0.48076923)
   2) THAL=Normal 140  34 N (0.75714286 0.24285714)
     4) CP=Atypical,NonAnginal,Typical 95  12 N (0.87368421 0.12631579) *
     5) CP=Asymptomatic 45  22 N (0.51111111 0.48888889)
      10) CA< 0.5 28   7 N (0.75000000 0.25000000) *
      11) CA>=0.5 17   2 Y (0.11764706 0.88235294) *
   3) THAL=Fixed.Defect,Reversible.Defect 120  29 Y (0.24166667 0.75833333)
     6) CA< 0.5 53  24 Y (0.45283019 0.54716981)
      12) EXANG=N 31  10 N (0.67741935 0.32258065)
        24) AGE>=51 20   3 N (0.85000000 0.15000000) *
        25) AGE< 51 11   4 Y (0.36363636 0.63636364) *
      13) EXANG=Y 22   3 Y (0.13636364 0.86363636) *
     7) CA>=0.5 67   5 Y (0.07462687 0.92537313) *
>
```

```
>
> # Fit a logistic regression model to the training data
> heart_train_logistic_regression_model_q2_7 <- glm(HD ~ ., data = heart_train_data, family = binomial)
> heart_train_logistic_regression_model_q2_7

Call:  glm(formula = HD ~ ., family = binomial, data = heart_train_data)

Coefficients:
          (Intercept)                   AGE                  SEXM            CPAtypical          CPNonAnginal
            -2.986357             -0.008904              1.468322            -0.989700            -1.740135
           CPTypical             TRESTBPS                  CHOL              FBS>120          RESTECGNormal
            -2.098109              0.024234              0.004740            -0.769233            -0.629625
     RESTECGST.T.Wave              THALACH                 EXANGY               OLDPEAK              SLOPEFlat
            -0.186629             -0.023936              1.029208              0.394286              1.200481
             SLOPEUp                    CA             THALNormal  THALReversible.Defect
             0.414334              1.300509              0.177427              1.437050

Degrees of Freedom: 259 Total (i.e. Null);  241 Residual
Null Deviance:      360.1
Residual Deviance: 170.9        AIC: 208.9
>
>
> # Make predictions on the test data
> heart_test_data_predictions <- predict(heart_train_logistic_regression_model_q2_7, newdata = heart_test_data, type = "respon
se")
> heart_test_data_predictions_tree <- predict(best_tree, newdata = heart_test_data)
>
> # Calculate prediction statistics using the provided function
> heart_test_data_prediction_stats <- my.pred.stats(heart_test_data_predictions, heart_test_data$HD)
------------------------------------------------------------------------------
Performance statistics:

Confusion matrix:

    target
pred  N  Y
   N 97 12
   Y 12 79

Classification accuracy = 0.88
Sensitivity             = 0.8681319
Specificity             = 0.8899083
Area-under-curve        = 0.9362839
Logarithmic loss        = 63.35378

------------------------------------------------------------------------------

------------------------------------------------------------------------------
> heart_test_data_prediction_stats_tree <- my.pred.stats(heart_test_data_predictions_tree[,2], heart_test_data$HD)
------------------------------------------------------------------------------
Performance statistics:

Confusion matrix:

    target
pred  N  Y
   N 96 11
   Y 13 80

Classification accuracy = 0.88
Sensitivity             = 0.8791209
Specificity             = 0.8807339
Area-under-curve        = 0.9058373
Logarithmic loss        = 70.55278

------------------------------------------------------------------------------
>
```

From the performance statistics displayed above, we can know that both prediction statistics for tree and stepwise logistic regression model shows that both classification accuracy is the same at 0.88.

The sensitivity of tree is 0. 8791209 but for the stepwise logistic regression model is 0. 8681319 which shows that the stepwise logistic regression has a higher sensitivity than the tree of 0.010989. This implies that the tree has a higher probability to detect true positive cases than the stepwise logistic regression.

The specificity of tree is 0.8807339 but for the stepwise logistic regression model is 0.8899083 which shows that the stepwise logistic regression has a higher specificity than the tree of 0.0091744. This implies that the stepwise logistic regression has a higher probability to detect true positive cases than the tree.

The area-under-curve of tree is 0.9058373 which is lower than the area-under-curve of stepwise logistic regression model by 0.0304466. The area-under-curve of stepwise logistic regression model is 0.9362839.

The logarithmic loss of tree is 70.55278 while the logarithmic loss of stepwise logistic regression model is 63.35378. The logarithmic loss of tree is 7.199 higher than the stepwise logistic regression model.

In conclusion, both tree and the stepwise logistic regression model has similar and close results of performance statistics. Either one would be equally preferable to be used as a diagnostic test.

Question 2.8

```
> ######################################################################
> # Assignment 3 Question 2.8a
> ######################################################################
> # Cross validation (test data)
> heart_test_data_cross_validation <- learn.tree.cv(HD ~ ., data = heart_test_data, nfolds = 10, m = 5000)
> heart_test_data_cross_validation
$best.tree
n= 200

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

 1) root 200 91 N (0.54500000 0.45500000)
   2) THAL=Normal 105 20 N (0.80952381 0.19047619)
     4) AGE< 58.5 72  6 N (0.91666667 0.08333333) *
     5) AGE>=58.5 33 14 N (0.57575758 0.42424242)
      10) CP=Atypical,NonAnginal,Typical 20  4 N (0.80000000 0.20000000) *
      11) CP=Asymptomatic 13  3 Y (0.23076923 0.76923077) *
   3) THAL=Fixed.Defect,Reversible.Defect 95 24 Y (0.25263158 0.74736842)
     6) CP=Atypical,NonAnginal,Typical 34 16 N (0.52941176 0.47058824)
      12) CA< 0.5 19  4 N (0.78947368 0.21052632) *
      13) CA>=0.5 15  3 Y (0.20000000 0.80000000) *
     7) CP=Asymptomatic 61  6 Y (0.09836066 0.90163934) *

$cv.stats
$cv.stats$cp
        1         2         3         4
0.51648352 0.06043956 0.03846154 0.01000000

$cv.stats$cv.err
        1         2         3         4
1.0000000 0.5400110 0.4523077 0.4079692

$cv.stats$n.leaves
[1] 1 2 4 6


$best.cp
   4
0.01

> plot.tree.cv(heart_test_data_cross_validation)
>
```

```
> # Get the decision tree (test data)
> heart_test_data_decision_tree_odds <- predict(best_tree, newdata = heart_test_data, type = "prob")[,2]
> heart_test_data_decision_tree_odds
        1         2         3         4         5         6         7         8         9        10        11        12
0.1500000 0.8823529 0.9253731 0.1263158 0.1263158 0.1263158 0.8823529 0.2500000 0.9253731 0.8636364 0.1500000 0.1263158
       13        14        15        16        17        18        19        20        21        22        23        24
0.9253731 0.6363636 0.1500000 0.1263158 0.6363636 0.2500000 0.1263158 0.1263158 0.1263158 0.1263158 0.1263158 0.9253731
       25        26        27        28        29        30        31        32        33        34        35        36
0.9253731 0.1263158 0.1263158 0.1263158 0.2500000 0.8636364 0.1263158 0.9253731 0.1263158 0.1500000 0.1263158 0.2500000
       37        38        39        40        41        42        43        44        45        46        47        48
0.8636364 0.9253731 0.9253731 0.1263158 0.9253731 0.8636364 0.1263158 0.1263158 0.2500000 0.9253731 0.1263158 0.6363636
       49        50        51        52        53        54        55        56        57        58        59        60
0.1263158 0.1263158 0.1263158 0.1500000 0.8823529 0.1263158 0.9253731 0.9253731 0.9253731 0.6363636 0.1263158 0.1263158
       61        62        63        64        65        66        67        68        69        70        71        72
0.8636364 0.1263158 0.9253731 0.1263158 0.9253731 0.9253731 0.1263158 0.1500000 0.8636364 0.1263158 0.1263158 0.9253731
       73        74        75        76        77        78        79        80        81        82        83        84
0.9253731 0.9253731 0.8823529 0.1263158 0.9253731 0.1263158 0.1263158 0.8636364 0.2500000 0.2500000 0.1263158 0.8636364
       85        86        87        88        89        90        91        92        93        94        95        96
0.1263158 0.1263158 0.1263158 0.1263158 0.2500000 0.1263158 0.2500000 0.9253731 0.9253731 0.1263158 0.1263158 0.9253731
       97        98        99       100       101       102       103       104       105       106       107       108
0.9253731 0.9253731 0.1263158 0.2500000 0.2500000 0.1263158 0.8823529 0.1263158 0.9253731 0.1500000 0.9253731 0.9253731
      109       110       111       112       113       114       115       116       117       118       119       120
0.9253731 0.6363636 0.8636364 0.8823529 0.1500000 0.8636364 0.9253731 0.6363636 0.1263158 0.2500000 0.9253731 0.9253731
      121       122       123       124       125       126       127       128       129       130       131       132
0.9253731 0.9253731 0.1263158 0.8636364 0.1263158 0.1263158 0.9253731 0.9253731 0.1263158 0.2500000 0.1500000 0.9253731

      133       134       135       136       137       138       139       140       141       142       143       144
0.1263158 0.2500000 0.1263158 0.1263158 0.8636364 0.9253731 0.8636364 0.1263158 0.1263158 0.1500000 0.1263158 0.8636364
      145       146       147       148       149       150       151       152       153       154       155       156
0.8636364 0.1263158 0.9253731 0.1263158 0.1263158 0.1263158 0.1500000 0.2500000 0.1500000 0.9253731 0.8823529 0.8823529
      157       158       159       160       161       162       163       164       165       166       167       168
0.8636364 0.9253731 0.9253731 0.9253731 0.6363636 0.8823529 0.1263158 0.2500000 0.1263158 0.8636364 0.1263158 0.1263158
      169       170       171       172       173       174       175       176       177       178       179       180
0.8636364 0.1263158 0.9253731 0.8636364 0.2500000 0.2500000 0.9253731 0.9253731 0.9253731 0.9253731 0.1263158 0.1263158
      181       182       183       184       185       186       187       188       189       190       191       192
0.6363636 0.9253731 0.1263158 0.1500000 0.2500000 0.1263158 0.6363636 0.9253731 0.9253731 0.9253731 0.1263158 0.9253731
      193       194       195       196       197       198       199       200
0.9253731 0.8823529 0.1263158 0.8823529 0.1263158 0.2500000 0.1263158 0.1263158
>
> # 69th patient in the test dataset odds
> heart_test_data_decision_tree_odds_69 <- heart_test_data_decision_tree_odds[69]
> heart_test_data_decision_tree_odds_69
       69
0.8636364
>

> ######################################################################
> # Assignment 3 Question 2.8b
> ######################################################################
> # Logistic regression model
> stepwise_selection_model_BIC_2_8b <- step(heart_train_logistic_regression_model, k = log(nrow(heart_train_data)), direction
 = "both")
> stepwise_selection_model_BIC_2_8b


Call:  glm(formula = HD ~ CP + THALACH + OLDPEAK + CA + THAL, family = binomial,
    data = heart_train_data)

Coefficients:
          (Intercept)             CPAtypical           CPNonAnginal               CPTypical                 THALACH
              2.74052               -1.18588               -1.89032               -1.85305               -0.02349
              OLDPEAK                     CA             THALNormal  THALReversible.Defect
              0.57627                1.09854               -0.32528                1.45941

Degrees of Freedom: 259 Total (i.e. Null);  251 Residual
Null Deviance:      360.1
Residual Deviance: 194.1        AIC: 212.1
>
> # Get the predicted probability of having heart disease ("Y") for the 69th patient in the test dataset
> prob_logistic_regression <- predict(stepwise_selection_model_BIC_2_8b, newdata = heart_test_data, type = "response")
> prob_logistic_regression
         1          2          3          4          5          6          7          8          9         10         11
0.21229679 0.98267202 0.99234444 0.13570556 0.11870564 0.07645869 0.94920922 0.25570083 0.93411572 0.91252546 0.37612977
        12         13         14         15         16         17         18         19         20         21         22
0.16575784 0.26084595 0.25916393 0.23002063 0.06655113 0.41178632 0.34249502 0.06752047 0.08000612 0.14386087 0.06491972
        23         24         25         26         27         28         29         30         31         32         33
0.18357419 0.90774162 0.99079631 0.09406169 0.02886170 0.35028877 0.32349776 0.93548744 0.56193821 0.96907634 0.03965615
```

```
        34         35         36         37         38         39         40         41         42         43         44
0.66943195 0.03077179 0.14596615 0.94382027 0.82542523 0.94729246 0.10741546 0.99547477 0.26343741 0.46281789 0.09608285
        45         46         47         48         49         50         51         52         53         54         55
0.17434297 0.72559606 0.11722089 0.93649697 0.16742286 0.08670354 0.16534191 0.75792796 0.47983007 0.03964219 0.93832287
        56         57         58         59         60         61         62         63         64         65         66
0.98211005 0.48112761 0.61962682 0.15984047 0.38481019 0.82568920 0.08111744 0.99661726 0.03020824 0.96924539 0.99075584
        67         68         69         70         71         72         73         74         75         76         77
0.19976429 0.34416615 0.94635095 0.29858162 0.07648459 0.93600541 0.99399076 0.82798064 0.34421501 0.07165247 0.97338832
        78         79         80         81         82         83         84         85         86         87         88
0.29980735 0.05293960 0.88626748 0.66083203 0.32875554 0.02295744 0.42742103 0.06319420 0.02403534 0.04148480 0.10185085
        89         90         91         92         93         94         95         96         97         98         99
0.20689997 0.06371670 0.28868760 0.99952978 0.96130028 0.03766343 0.02886170 0.81995171 0.93425186 0.98532492 0.28433488
       100        101        102        103        104        105        106        107        108        109        110
0.12405875 0.12663437 0.02859066 0.44480479 0.19299084 0.97046392 0.34278004 0.81645725 0.52862775 0.98343792 0.83234873
       111        112        113        114        115        116        117        118        119        120        121
0.79349975 0.69471082 0.02721790 0.93898142 0.86066233 0.17559230 0.03384610 0.25849642 0.99564333 0.98070341 0.94649470
       122        123        124        125        126        127        128        129        130        131        132
0.99793922 0.10497544 0.99199176 0.16515523 0.07335651 0.99623200 0.98114302 0.05927129 0.19557335 0.28630062 0.44773075
       133        134        135        136        137        138        139        140        141        142        143
0.02885142 0.12405875 0.03782426 0.14851235 0.94056173 0.92410896 0.88773451 0.12005013 0.06763688 0.21868921 0.04337874
       144        145        146        147        148        149        150        151        152        153        154
0.56690819 0.27635383 0.04538682 0.99428322 0.02459264 0.05927129 0.10569437 0.24169089 0.47372341 0.37114318 0.91316669
       155        156        157        158        159        160        161        162        163        164        165
0.92595740 0.98936695 0.74222927 0.91526012 0.95667086 0.60752066 0.34278004 0.87044598 0.09406169 0.53128482 0.19950969
       166        167        168        169        170        171        172        173        174        175        176
0.56292215 0.03090414 0.19661832 0.63063722 0.11787187 0.92041885 0.83091936 0.28002751 0.35853810 0.95207399 0.98059372
       177        178        179        180        181        182        183        184        185        186        187
0.98369153 0.92972012 0.25207458 0.43934063 0.64294049 0.98143776 0.27656658 0.79586519 0.20307141 0.31455734 0.14342795
       188        189        190        191        192        193        194        195        196        197        198
0.88401669 0.38493857 0.96536720 0.03541671 0.99913304 0.88043218 0.98681556 0.21208079 0.89973363 0.20435627 0.26105099
       199        200
0.12535263 0.08513887
>
> prob_logistic_regression_69 <- prob_logistic_regression[69]
> prob_logistic_regression_69
        69
0.9463509
>
```

$$Equation\ to\ calculate\ Odds\colon\ O = \frac{P(Y=1)}{P(Y=0)}$$

For question 8(a), the odds of having heart disease for the 69th patient using the tree model found using cross validation is 0.8636364.

$$O = \frac{0.8636364}{0.1363636} = 6.333$$

For question 8(b), the odds of having heart disease for the 69th patient using the stepwise logistic regression model is 0.9463509.

$$O = \frac{0.9463509}{1-0.9463509} = 17.6369$$

The odds predicted by stepwise logistic regression is 0.0827145 higher than the odds predicted by using the tree model.

Question 2.9

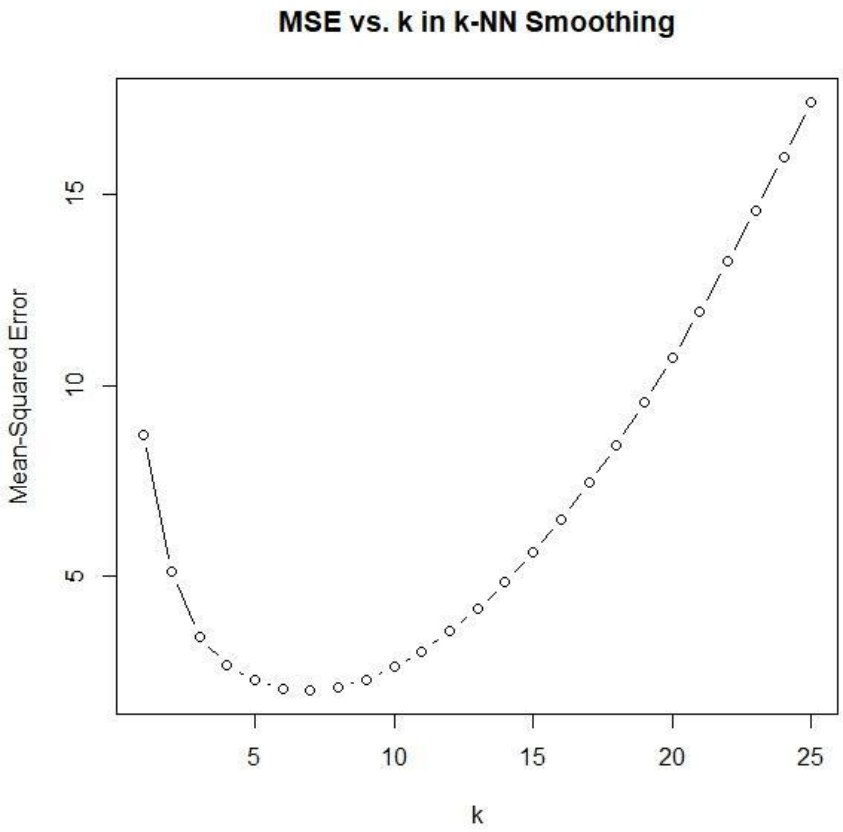
```
> ##################################################################
> # Assignment 3 Question 2.9
> ##################################################################
> # Load the Bootstrap package
> library(boot)
>
> # Define the modified boot.auc function from Studio10
> boot_prob_heart_disease <- function(formula, heart_test_data, indices) {
+   # Create a bootstrapped version of heart_test_data
+   booted = heart_test_data[indices, ]
+   # Fit a logistic regression to the bootstrapped data
+   booted_logistic_regresion = glm(formula, booted, family = binomial)
+   # Calculate the predicted probabilities for the 69th patient
+   predicted_probability = predict(booted_logistic_regresion, newdata = heart_test_data[69, ], type = "response")
+
+   return(predicted_probability)
+ }
>
> bootstrap <- boot(data = heart_test_data, statistic = boot_prob_heart_disease, R = 5000, formula = HD ~ .)
There were 50 or more warnings (use warnings() to see the first 50)
> bootstrap

ORDINARY NONPARAMETRIC BOOTSTRAP


Call:
boot(data = heart_test_data, statistic = boot_prob_heart_disease,
    R = 5000, formula = HD ~ .)


Bootstrap Statistics :
     original        bias    std. error
t1* 0.9939895 -0.0005310357   0.0141703
>
> boot.ci(bootstrap, conf = 0.95, type = "bca")
BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 5000 bootstrap replicates

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

Intervals :
Level       BCa
95%   ( 0.7095,  0.9997 )
Calculations and Intervals on Original Scale
Some BCa intervals may be unstable
>
```

The confidence interval obtained from bootstrapping is 0.7095 to 0.9997 where 0.7095 is the lower bound and 0.9997 is the upper bound. The predicted probability of having heart disease for the 69th patient using the tree model found using cross validation is 0.8636364 while the odds of having heart disease for the 69th patient using the stepwise logistic regression model is 0.9463509. Both 0.8636364 and 0.9463509 lies within the confidence interval so this shows that the prediction is confident to be correct and may be accurate.

Question 3

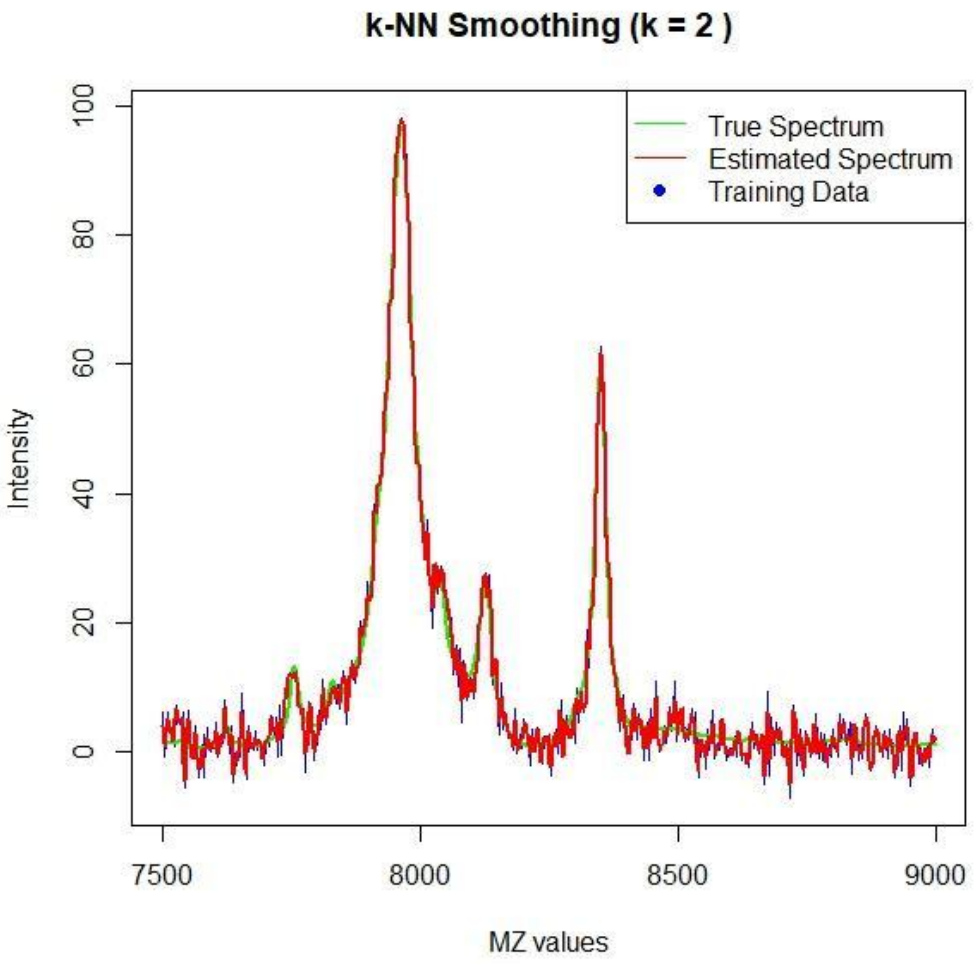
Question 3.1

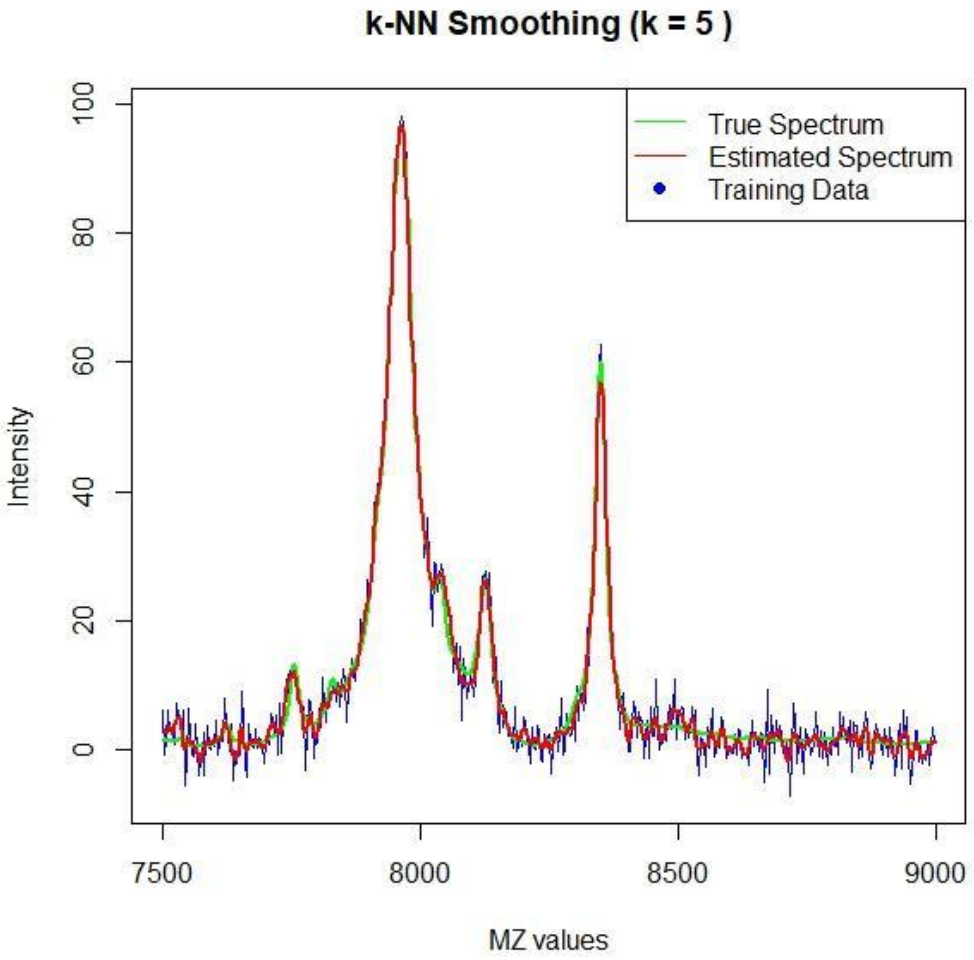
```
> ######################################################################
> # Assignment 3 Question 3.1
> ######################################################################
> # Load the kknn package
> library(kknn)
> # Load the Bootstrap package
> library(boot)
>
> # Load the given data
> ms_measured <- read.csv("ms.measured.2023.csv")
> ms_truth <- read.csv("ms.truth.2023.csv")
> mean_square_error <- numeric(length = 25)
>
> # Loop through k from 1 to 25
> for (k in 1:25) {
+   # Fit k-NN model
+   knn_model <- fitted(kknn(intensity ~ ., ms_measured, ms_truth, k = k, kernel = "optimal"))
+   # Compute the mean-squared error
+   mse <- mean((knn_model - ms_truth$intensity)^2)
+   # Store the MSE value for this value of k
+   mean_square_error[k] <- mse
+ }
>
> # Create a plot of MSE values against k
> plot(1:25, mean_square_error, type = "b", xlab = "k", ylab = "Mean-Squared Error", main = "MSE vs. k in k-NN Smoothing")
>
```

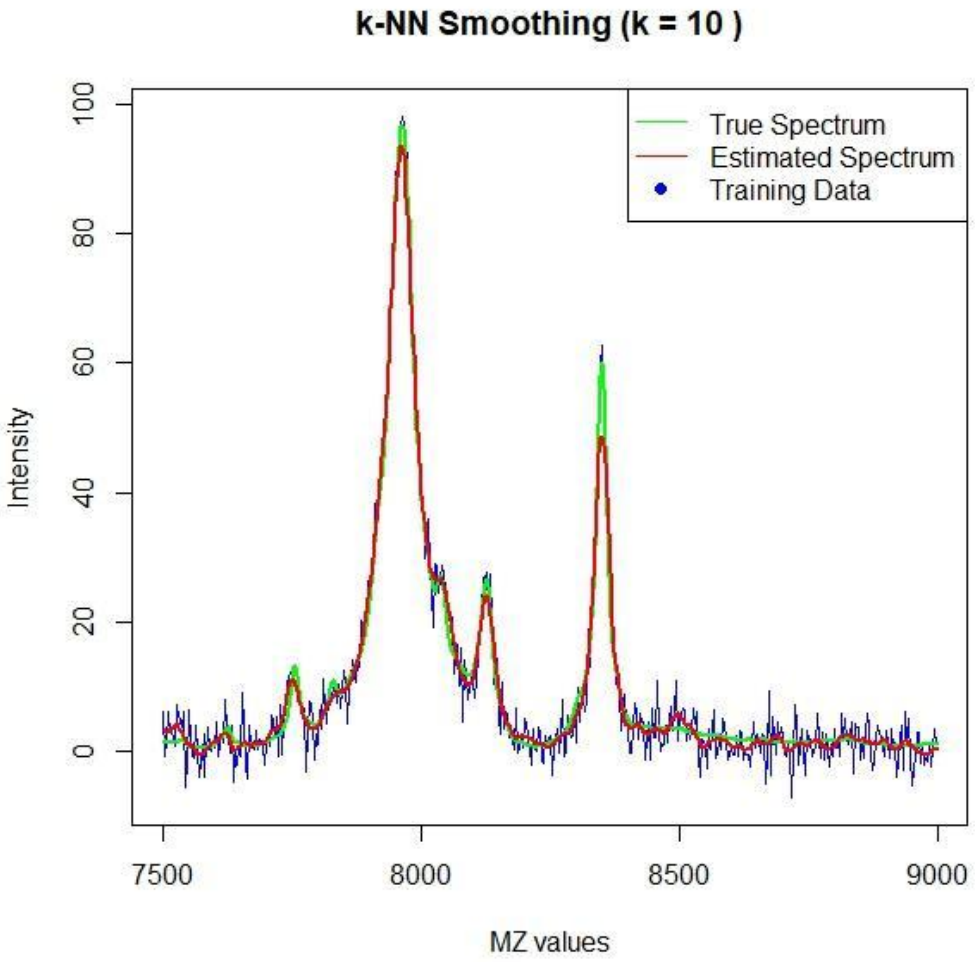


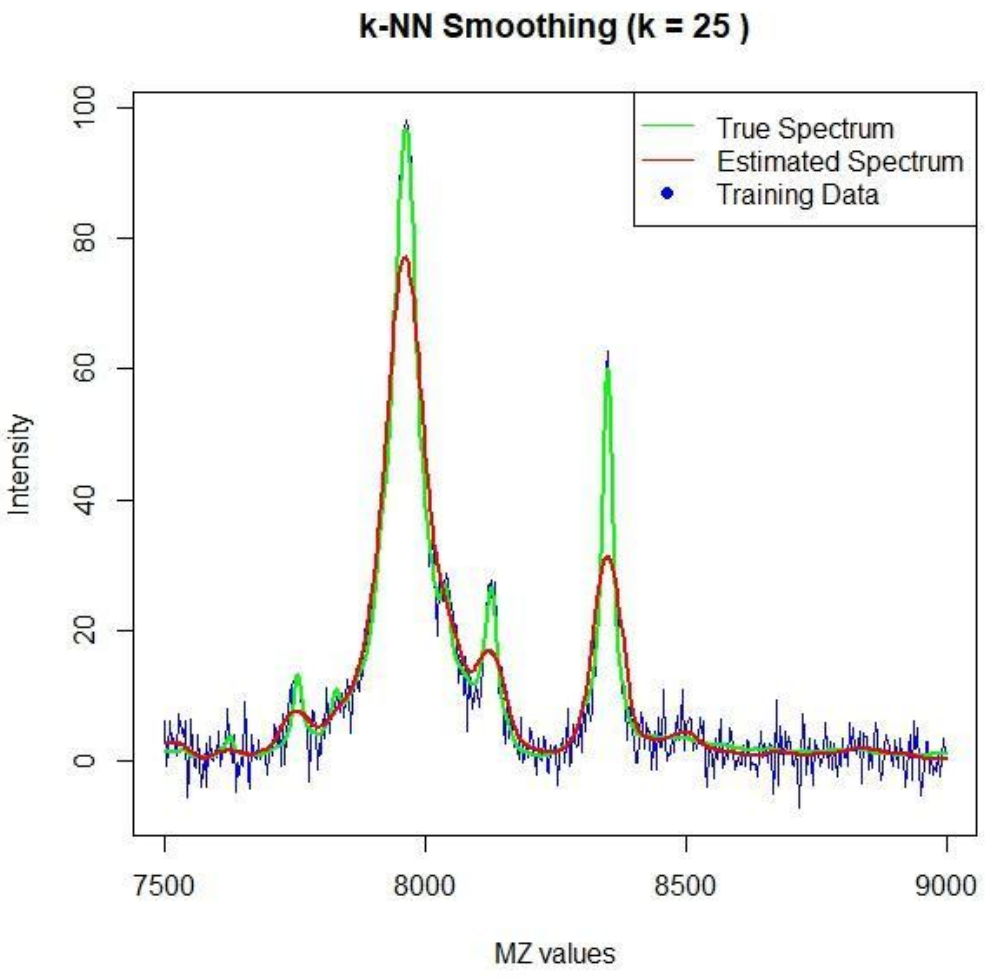
Question 3.2

```
> ######################################################################
> # Assignment 3 Question 3.2
> ######################################################################
> # (i) the training data points (ms_measured$intensity)
> # (ii) the true spectrum (ms_truth$intensity)
> # (iii) the estimated spectrum (predicted intensity values for the MZ values
> # in ms_truth) produced by the k-NN method for different k values
```




k-NN Smoothing (k = 5)
True Spectrum
Estimated Spectrum
Training Data
Intensity
0
20
40
60
80
100
7500
8000
8500
9000
MZ values


k-NN Smoothing (k = 10)
True Spectrum
Estimated Spectrum
Training Data
Intensity
0
20
40
60
80
100
7500
8000
8500
9000
MZ values



Red Line is the ms_measured$intensity which is the training data points.

Green Line is the ms_truth$intensity which is the true spectrum.

Blue Line is the predicted intensity values for the MZ values in ms_truth which is the estimated spectrum.

Question 3.3

```
> ######################################################################
> # Assignment 3 Question 3.3
> ######################################################################
> k_value = 2
> sqrt(mean((fitted(kknn(intensity ~ ., ms_measured, ms_truth, k = k_value, kernel = "optimal"))-ms_truth$intensity)^2))
[1] 2.259376
>
> k_value = 5
> sqrt(mean((fitted(kknn(intensity ~ ., ms_measured, ms_truth, k = k_value, kernel = "optimal"))-ms_truth$intensity)^2))
[1] 1.504264
>
> k_value = 10
> sqrt(mean((fitted(kknn(intensity ~ ., ms_measured, ms_truth, k = k_value, kernel = "optimal"))-ms_truth$intensity)^2))
[1] 1.615091
>
> k_value = 25
> sqrt(mean((fitted(kknn(intensity ~ ., ms_measured, ms_truth, k = k_value, kernel = "optimal"))-ms_truth$intensity)^2))
[1] 4.17383
>
```

Qualitatively, the graph when k=5 is the where the lines overlap each other the most which implies that the true values is very close to the estimated spectrum unlike when k=2 where although the lines still overlap each other but it looks very forcefully in which it shows signs of overfitting. When k=10, the peak for the red line is slightly lowered which already show signs of underfitting but the rest of the data is still closely placed together which also implies that the true values it is still close to the estimated spectrum but not as close as when k=5. When k=25, the peak or the red line is

lowered greatly and the rest of the data is not closely placed together which implies that the true value is far from the estimated spectrum and hence the data is underfitted.

Quantitatively, from all these k values of 2, 5, 10 and 25, with the mean square error (MSE) on true spectrum calculated, the most optimal k-nn model is at k=5 as its mean square error is 1.504264 which is the lowest amongst the k values of 2, 5, 10 and 25.

Question 3.4

```
> ###################################################################
> # Assignment 3 Question 3.4
> ###################################################################
> # No code
>
```

The estimated spectra that provide a smooth, low-noise estimation of background levels as well as accurate estimation of the peak would be indicated but the values obtained from question 3.3 whereby a lower MSE value indicated that it would be a closer match between the estimated and true values and from question 3.3 the lowest MSE value is when k=5. The MSE value when k=10 is close to the MSE value when k=5 with a difference of 0.110827. This implies that k values between 5 to 10 could contain the optimal value of k that would fit this dataset.

In my opinion, the estimated spectra when k=5 does achieve the aim of providing a smooth, low-noise estimation of background levels as well as accurate estimation of the peak to a certain extend as visually, the graph looks a bit overfitted.

I think the k-NN method may not be an appropriate method to achieve the aim of providing a smooth, low-noise estimation of background levels as well as accurate estimation of the peak unless the optimal k value that fits the dataset is found.

Question 3.5

```
> ###################################################################
> # Assignment 3 Question 3.5
> ###################################################################
> # Perform cross-validation to find the best k
> knn_cross_validate <- train.kknn(intensity ~ ., ms_measured, kmax = 25, kernel = "optimal")
> best_k <- knn_cross_validate$best.parameters$k
> best_k
[1] 6
>
```

The optimal k value would be when k=6. Comparing to the value of k that would minimised the actual MSE as computed in question3.1 as shown in the image below, when k=7, the MSE=2.004127 and when k=6, the MSE=2.021296, both values is quite close to each other with a difference of only 0.017169.

```
> mean_square_error
 [1]  8.704256  5.104779  3.410489  2.656165  2.262812  2.021296  2.004127  2.084660  2.286621  2.608518  3.012139
[12]  3.553871  4.124015  4.838148  5.619558  6.482609  7.436011  8.422623  9.547819 10.733335 11.927679 13.234540
[23] 14.597129 15.985650 17.420855
>
```

Question 3.6

```
> ###################################################################
> # Assignment 3 Question 3.6
> ###################################################################
> knn_model_best <- fitted(kknn(intensity ~ MZ, ms_measured, ms_truth, k = best_k, kernel = "optimal"))
> knn_model_best_noise <- ms_truth - knn_model_best
>
> sqrt(var(knn_model_best_noise))
                MZ intensity
MZ        437.83939  9.925710
intensity   9.92571  1.421134
>
```

The estimated of the standard deviation of the noise that corrupted the intensity measurement is 1.421134.

Question 3.7

```
> ############################################################
> # Assignment 3 Question 3.7
> ############################################################
> # Find the index of the maximum estimated abundance
> index_of_max_abundance <- which.max(knn_model_best)
> index_of_max_abundance
[1] 283
> # Find the MZ value corresponding to the maximum estimated abundance
> MZ_of_max_abundance <- ms_truth$MZ[index_of_max_abundance]
> MZ_of_max_abundance
[1] 7963.3
>
```

The value that MZ corresponds to the maximum estimated abundance is 7963.3 as displayed in the diagram above.

Question 3.8

```
> ################################################################
> # Assignment 3 Question 3.8
> ################################################################
> # Load the Bootstrap package
> library(boot)
>
> # 3.5 best_k
> best_k_neighbours <- fitted(kknn(intensity~., ms_measured, ms_truth, kernel = 'optimal', k = best_k))
> boot_max_best_k = function(formula, ms_truth, indices){
+   # Create a bootstrapped version of ms_truth
+   booted = ms_truth[indices, ]
+   # Target = best_k
+   target = ms_truth[which.max(best_k_neighbours),]
+   mz = fitted(kknn(formula, booted, target, kernel = 'optimal', k = best_k))
+   return(mz)
+ }
>
> # best_k bootstrap
> bootstrap_best_k <- boot(data = ms_truth, statistic = boot_max_best_k, R = 5000, formula = intensity ~ .)
> bootstrap_best_k

ORDINARY NONPARAMETRIC BOOTSTRAP


Call:
boot(data = ms_truth, statistic = boot_max_best_k, R = 5000,
    formula = intensity ~ .)


Bootstrap Statistics :
    original     bias    std. error
t1* 96.36757 -0.2812708   0.6585319
>
> # best_k confidence interval
> boot.ci(bootstrap_best_k, conf = 0.95, type = "bca")
BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 5000 bootstrap replicates

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

Intervals :
Level       BCa
95%   (95.23, 96.63 )
Calculations and Intervals on Original Scale
>
> # k = 3 neighbours
> k_three <- 3
> three_k_neighbours <- fitted(kknn(intensity~., ms_measured, ms_truth, kernel = 'optimal', k = k_three))
> boot_max_k_three = function(formula, ms_truth, indices){
+   # Create a bootstrapped version of ms_truth
+   booted = ms_truth[indices, ]
+   # Target = k_three
+   target = ms_truth[which.max(three_k_neighbours),]
+   mz = fitted(kknn(formula, booted, target, kernel = 'optimal', k = k_three))
+   return(mz)
+ }
>
> # k_three bootstrap
> bootstrap_k_three <- boot(data = ms_truth, statistic = boot_max_k_three, R = 5000, formula = intensity ~ .)
> bootstrap_k_three
```

```
ORDINARY NONPARAMETRIC BOOTSTRAP


Call:
boot(data = ms_truth, statistic = boot_max_k_three, R = 5000,
    formula = intensity ~ .)


Bootstrap Statistics :
    original     bias    std. error
t1*  96.5963 -0.1441236   0.3407716
>
> # k_three confidence interval
> boot.ci(bootstrap_k_three, conf = 0.95, type = "bca")
BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 5000 bootstrap replicates

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

Intervals :
Level       BCa
95%   (95.85, 96.64 )
Calculations and Intervals on Original Scale
>
```

```
> # k = 20 neighbours
> k_twenty <- 20
> twenty_k_neighbours <- fitted(kknn(intensity~., ms_measured, ms_truth, kernel = 'optimal', k = k_twenty))
> boot_max_k_twenty = function(formula, ms_truth, indices){
+   # Create a bootstrapped version of ms_truth
+   booted = ms_truth[indices, ]
+   # Target = k_twenty
+   target = ms_truth[which.max(twenty_k_neighbours),]
+   mz = fitted(kknn(formula, booted, target, kernel = 'optimal', k = k_twenty))
+   return(mz)
+ }
>
> # best_k bootstrap
> bootstrap_k_twenty <- boot(data = ms_truth, statistic = boot_max_k_twenty, R = 5000, formula = intensity ~ .)
> bootstrap_k_twenty

ORDINARY NONPARAMETRIC BOOTSTRAP


Call:
boot(data = ms_truth, statistic = boot_max_k_twenty, R = 5000,
    formula = intensity ~ .)


Bootstrap Statistics :
    original     bias    std. error
t1* 92.26351 -0.7144465    2.572743
>
> # best_k confidence interval
> boot.ci(bootstrap_k_twenty, conf = 0.95, type = "bca")
BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 5000 bootstrap replicates

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

Intervals :
Level       BCa
95%   (86.67, 95.31 )
Calculations and Intervals on Original Scale
>
```

The confidence interval for the optimal k value (k=6) is (95.23, 96.63).

The confidence interval when k=3 is (95.85, 96.64).

The confidence interval when k=20 is (86.67, 95.31).

From the confidence interval obtained for different values of k as displayed above, we can see that the larger the k value is the larger the confidence interval is. When k=3, the difference between the upper and lower bound of the confidence interval is 0.79 but when k=20, the difference between the upper and lower bound of the confidence interval is 8.64 and 8.64 is bigger than 0.79 by 7.85. This is mostly due to when there is more neighbours involved, like how k=20 have more neighbours than when k=3, the prediction is less accurate and precise.