Heart Disease Prediction using Decision Tree

Kyle Anthony F. Niosco
III – BSCS STDS

I. INTRODUCTION

To prevent diseases from getting worse, early detection and prevention are vital. As a result, an immediate remedy to this is necessary. The application of machine learning, especially in predictive analytics, is one potential solution for this issue. Making precise decisions based on the information at hand is made easier with the help of predictive analytics. In this way, medical issues that demand human competence and have time limits can be handled by machine learning [1].

In this project, the decision tree, a classification machine learning technique, was used to predict the presence or absence of heart disease in a person. For the implementation, a dataset by Puspita Saha was extracted from the Kaggle Dataset Repository and examined using the aforementioned algorithm. The Decision Tree algorithm was used in the particular dataset since it is claimed to be a preferred method for predictive modeling because it is both simple to learn and highly effective. [2]. In addition, because they clearly lay out the problem so that all options can be considered, decision trees allow us to fully analyze the potential consequences of a decision, provide a framework to quantify the values of outcomes and the probabilities of achieving them, and help us to make the best decisions based on the available information and best assumptions [3].

II. OBJECTIVES OF THE STUDY

This paper seeks to predict decisions about the presence or absence of a heart disease in a person. In line with this, the following are the objectives:

- To prepare the Heart Disease Prediction dataset through cleaning and wrangling, to fit in the model
- To analyze the Heart Disease Prediction dataset using the Decision Tree algorithm
- To evaluate the results of the Decision Tree model generated

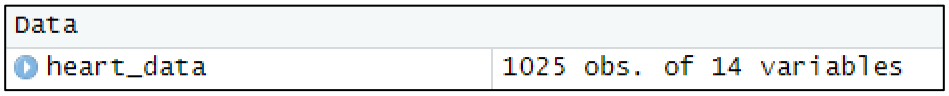
III. DATA

The data set used is about Heart Disease Prediction by Puspita Saha, obtained from Kaggle Dataset Repository: *https://www.kaggle.com/datasets/puspitasaha/heart-disease-prediction* [4]. The dataset contains 1025 rows (instances), and 14 columns (attributes).

Attributes	Description
age	Age in years
sex	1 = male; 0 = female
cp	Chest pain type
trestbps	Resting blood pressure
chol	Serum cholestoral in mg/dl
fbs	Fasting blood sugar
restecg	Resting electrocardiographic results
thalach	Maximum heart rate achieved
exang	Exercise induced angina
oldpeak	ST depression induced by exercise relative to rest
slope	Slope of peak exercise ST segment
ca	Number of major vessels (0-3)
thal	Thalassemia: 0 = null; 1 = fixed defect; 2 = normal blood flow; 3 = reversable defect
target	0 = no heart disease; 1 = with heart disesase

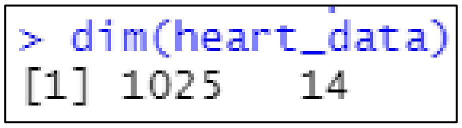
To implement the algorithm, R programming was the language used, while the Integrated Development Environment (IDE) used was RStudio. In RStudio, the dataset was loaded as a .csv file, and was explored using the following syntax below, together with its outputs:

- heart_data <- read.csv("heart (1).csv") — a variable is declared that will be used to open/read/load the dataset. The result was shows that it was loaded to the environment pane/window, which states that the data consists of 1025 observations of 14 variables.



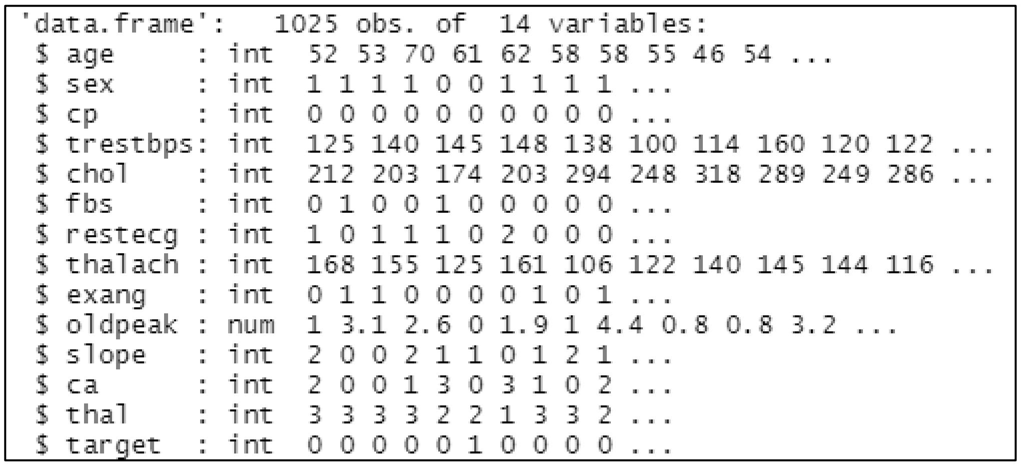
```
Data
heart_data                      1025 obs. of 14 variables
```

- dim(heart_data) — the dim() function was called to display the dimension of the dataset, wherein, according to the result, there are 1025 rows and 14 columns.



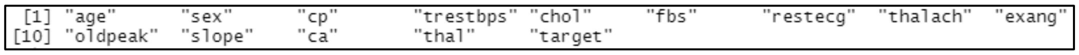
```
> dim(heart_data)
[1] 1025   14
```

- str(heart_data) — the str() function was called to display the internal structure of the dataset and to get to know about the particular object. The column names with the first 10 values of each are produced.



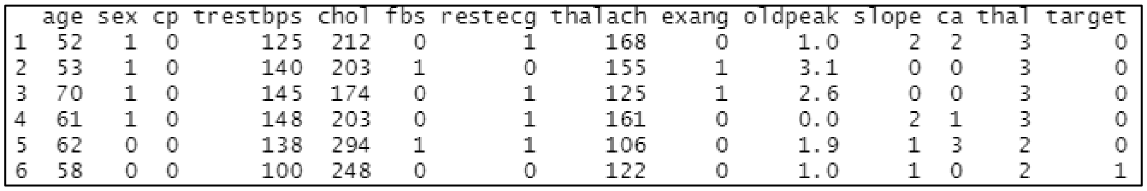
```
'data.frame':   1025 obs. of  14 variables:
 $ age     : int  52 53 70 61 62 58 58 55 46 54 ...
 $ sex     : int  1 1 1 1 0 0 1 1 1 1 ...
 $ cp      : int  0 0 0 0 0 0 0 0 0 0 ...
 $ trestbps: int  125 140 145 148 138 100 114 160 120 122 ...
 $ chol    : int  212 203 174 203 294 248 318 289 249 286 ...
 $ fbs     : int  0 1 0 0 1 0 0 0 0 0 ...
 $ restecg : int  1 0 1 1 1 0 2 0 0 0 ...
 $ thalach : int  168 155 125 161 106 122 140 145 144 116 ...
 $ exang   : int  0 1 1 0 0 0 0 1 0 1 ...
 $ oldpeak : num  1 3.1 2.6 0 1.9 1 4.4 0.8 0.8 3.2 ...
 $ slope   : int  2 0 0 2 1 1 0 1 2 1 ...
 $ ca      : int  2 0 0 1 3 0 3 1 0 2 ...
 $ thal    : int  3 3 3 3 2 2 1 3 3 2 ...
 $ target  : int  0 0 0 0 0 1 0 0 0 0 ...
```

- colnames(heart_data) — the colnames() function was called to know what are the all the column names inside the dataset.



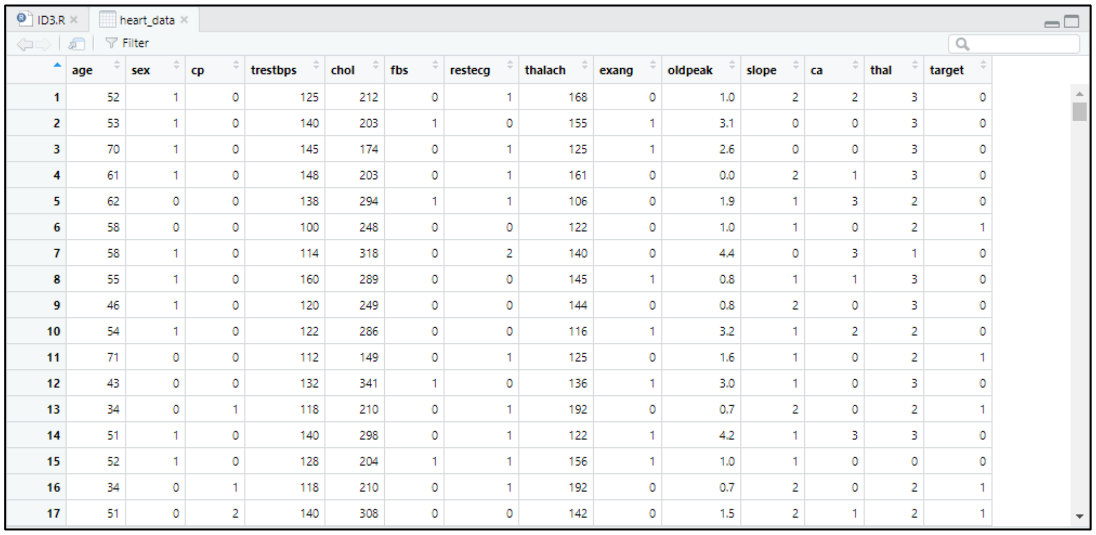
```
 [1] "age"      "sex"      "cp"       "trestbps" "chol"     "fbs"      "restecg"  "thalach"  "exang"
[10] "oldpeak"  "slope"    "ca"       "thal"     "target"
```

- head(heart_data) — the head() function was called to display the first 5 values of the dataset.



```
  age sex cp trestbps chol fbs restecg thalach exang oldpeak slope ca thal target
1  52   1  0      125  212   0       1     168     0     1.0     2  2    3      0
2  53   1  0      140  203   1       0     155     1     3.1     0  0    3      0
3  70   1  0      145  174   0       1     125     1     2.6     0  0    3      0
4  61   1  0      148  203   0       1     161     0     0.0     2  1    3      0
5  62   0  0      138  294   1       1     106     0     1.9     1  3    2      0
6  58   0  0      100  248   0       0     122     0     1.0     1  0    2      1
```

- View(heart_data) — the View() function was called to be able to see a full view of the dataset in the IDE used.



	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
1	52	1	0	125	212	0	1	168	0	1.0	2	2	3	0
2	53	1	0	140	203	1	0	155	1	3.1	0	0	3	0
3	70	1	0	145	174	0	1	125	1	2.6	0	0	3	0
4	61	1	0	148	203	0	1	161	0	0.0	2	1	3	0
5	62	0	0	138	294	1	1	106	0	1.9	1	3	2	0
6	58	0	0	100	248	0	0	122	0	1.0	1	0	2	1
7	58	1	0	114	318	0	2	140	0	4.4	0	3	1	0
8	55	1	0	160	289	0	0	145	1	0.8	1	1	3	0
9	46	1	0	120	249	0	0	144	0	0.8	2	0	3	0
10	54	1	0	122	286	0	0	116	1	3.2	1	2	2	0
11	71	0	0	112	149	0	1	125	0	1.6	1	0	2	1
12	43	0	0	132	341	1	0	136	1	3.0	1	0	3	0
13	34	0	1	118	210	0	1	192	0	0.7	2	0	2	1
14	51	1	0	140	298	0	1	122	1	4.2	1	3	3	0
15	52	1	0	128	204	1	1	156	1	1.0	1	0	0	0
16	34	0	1	118	210	0	1	192	0	0.7	2	0	2	1
17	51	0	2	140	308	0	0	142	0	1.5	2	1	2	1

IV. ALGORITHM

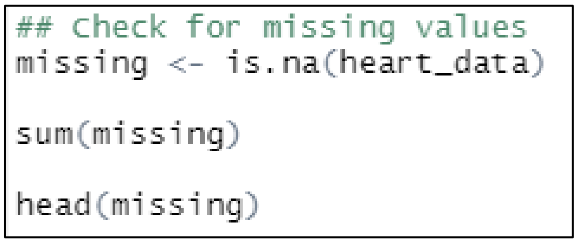
In implementing machine learning algorithms in R, it is important to import and load the necessary packages. Only the **tree** package was used for this project because that is what is used for classification and regression trees.

```
## Importing and Loading Libraries
library(tree)
```

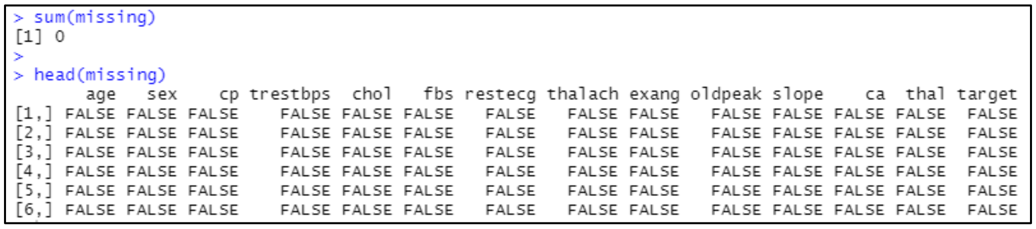
After importing and loading the necessary package, the dataset was loaded and its information was explored. The results of the data exploration were already shown above in the Data (Section 3).

```
## Opening/Reading the Dataset
# Heart Disease Prediction by PUSPITA SAHA
# https://www.kaggle.com/datasets/puspitasaha/heart-disease-prediction

heart_data <- read.csv("heart (1).csv")

## Data Exploration
dim(heart_data)

str(heart_data)

colnames(heart_data)

head(heart_data)

View(heart_data)
```

Part of the cleaning part that was conducted using the dataset was checking if there are missing values or null values. Based from the particular code block executed, there were no missing values found.

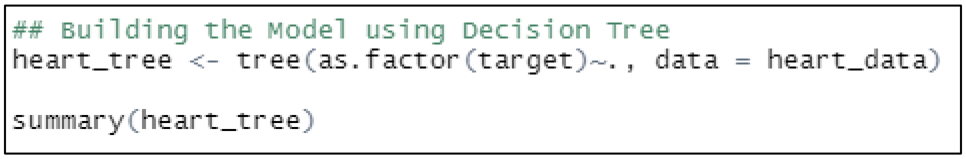


```
## Check for missing values
missing <- is.na(heart_data)

sum(missing)

head(missing)
```

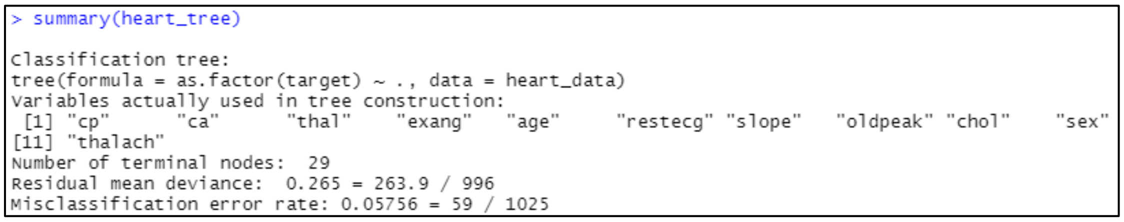


```
> sum(missing)
[1] 0
>
> head(missing)
       age   sex    cp trestbps  chol   fbs restecg thalach exang oldpeak slope    ca  thal target
[1,] FALSE FALSE FALSE    FALSE FALSE FALSE   FALSE   FALSE FALSE   FALSE FALSE FALSE FALSE  FALSE
[2,] FALSE FALSE FALSE    FALSE FALSE FALSE   FALSE   FALSE FALSE   FALSE FALSE FALSE FALSE  FALSE
[3,] FALSE FALSE FALSE    FALSE FALSE FALSE   FALSE   FALSE FALSE   FALSE FALSE FALSE FALSE  FALSE
[4,] FALSE FALSE FALSE    FALSE FALSE FALSE   FALSE   FALSE FALSE   FALSE FALSE FALSE FALSE  FALSE
[5,] FALSE FALSE FALSE    FALSE FALSE FALSE   FALSE   FALSE FALSE   FALSE FALSE FALSE FALSE  FALSE
[6,] FALSE FALSE FALSE    FALSE FALSE FALSE   FALSE   FALSE FALSE   FALSE FALSE FALSE FALSE  FALSE
```

One of the integral parts of this algorithm is to build a model using Decision Tree. In model creation, variables will be selected as part of the tree construction. To produce the number of terminal nodes, residual mean deviance and misclassification error rate, the summary() function was called.

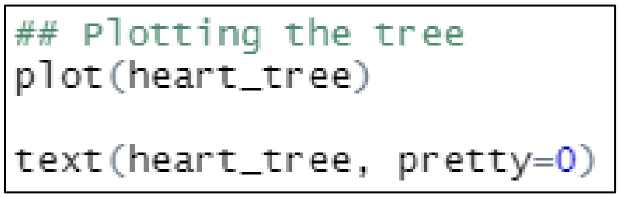


```
## Building the Model using Decision Tree
heart_tree <- tree(as.factor(target)~., data = heart_data)

summary(heart_tree)
```

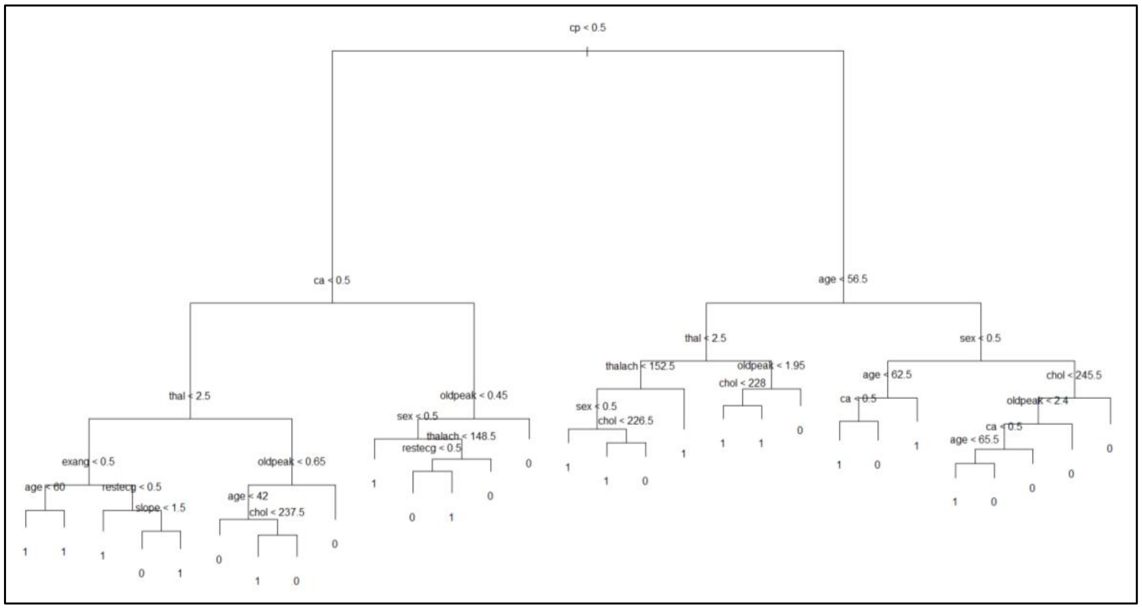


```
> summary(heart_tree)

Classification tree:
tree(formula = as.factor(target) ~ ., data = heart_data)
Variables actually used in tree construction:
 [1] "cp"      "ca"      "thal"    "exang"   "age"     "restecg" "slope"   "oldpeak" "chol"    "sex"
[11] "thalach"
Number of terminal nodes:  29
Residual mean deviance:  0.265 = 263.9 / 996
Misclassification error rate: 0.05756 = 59 / 1025
```

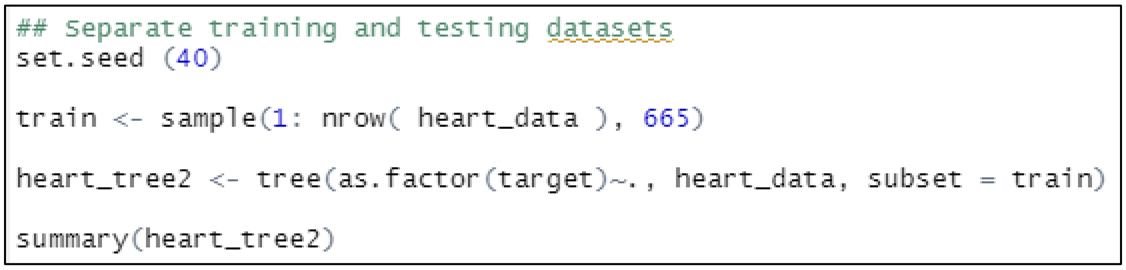
After building the model using decision tree, it was then initially visualized by plotting it. Here, the plot() function was called.



```
## Plotting the tree
plot(heart_tree)

text(heart_tree, pretty=0)
```

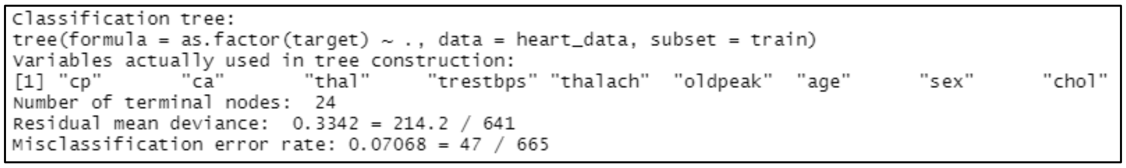


Another important part of the implementation is the separation or the split of the dataset into training and testing sets as they will be used to evaluate the model. Sixty-five percent (65%) of the dataset was assigned to training set while the remaining 35% was allocated to the testing set.



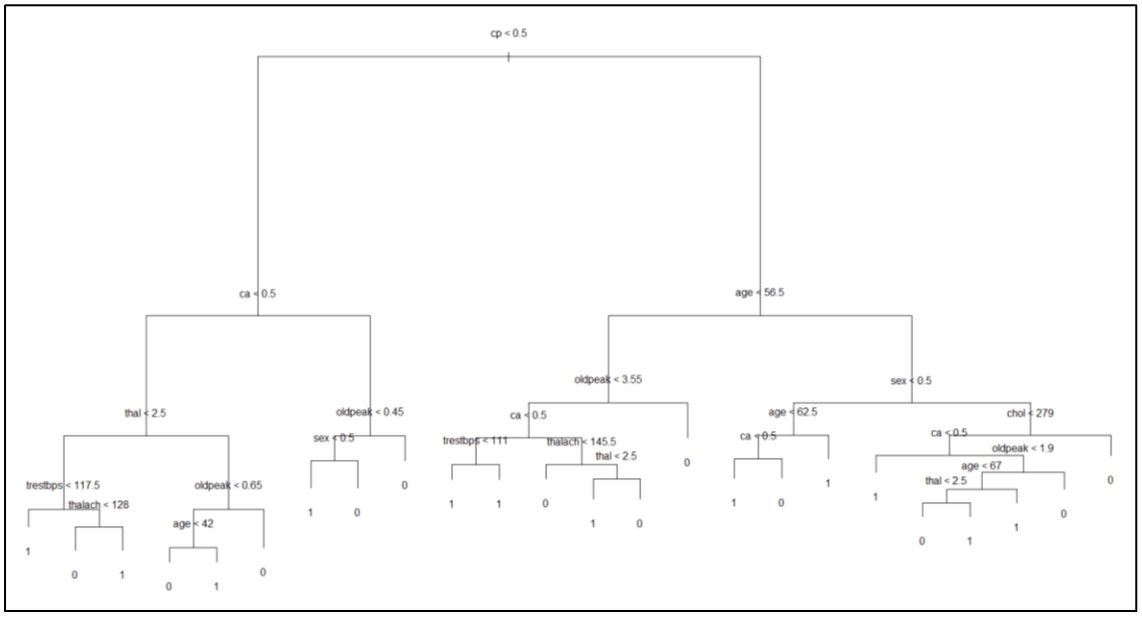
```
## Separate training and testing datasets
set.seed (40)

train <- sample(1: nrow( heart_data ), 665)

heart_tree2 <- tree(as.factor(target)~., heart_data, subset = train)

summary(heart_tree2)
```

The summary() function was called again to check again what variables are used in the tree construction, the number of terminal nodes, residual mean deviance, and misclassification rate. The function was also called to see the changes after the dataset was split into training and testing sets.



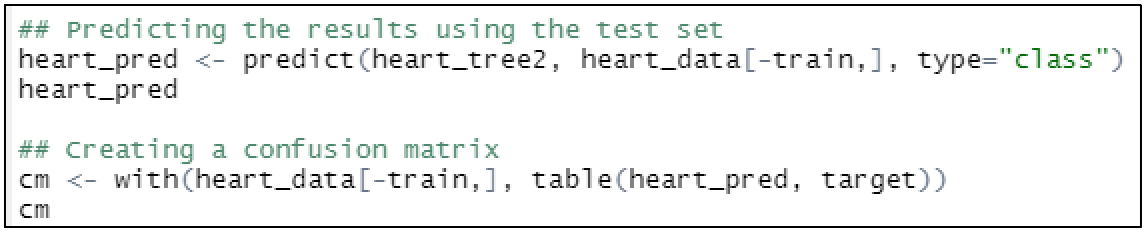
```
Classification tree:
tree(formula = as.factor(target) ~ ., data = heart_data, subset = train)
Variables actually used in tree construction:
[1] "cp"       "ca"       "thal"     "trestbps" "thalach"  "oldpeak"  "age"      "sex"      "chol"
Number of terminal nodes:  24
Residual mean deviance:  0.3342 = 214.2 / 641
Misclassification error rate: 0.07068 = 47 / 665
```

To visualize the new tree after the training and testing sets were split and the terminal nodes were reduced, the plot() function was called again.

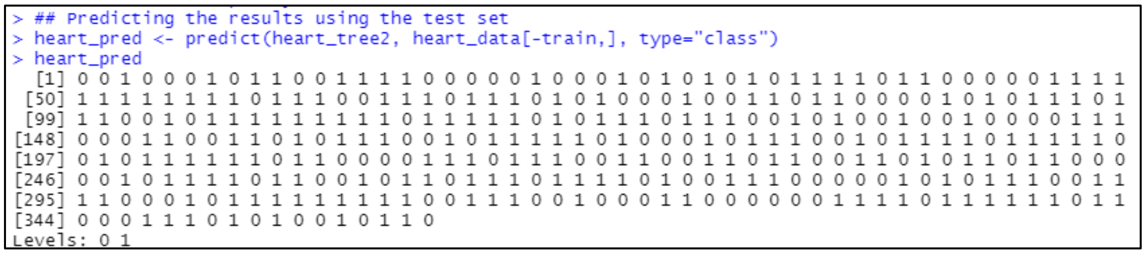


V. RESULTS AND DISCUSSION

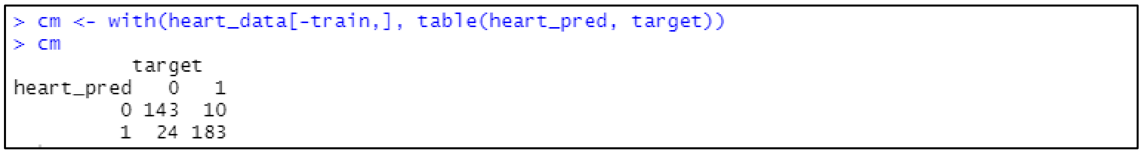
After splitting the dataset into training and testing sets, the prediction was conducted using the testing set. A confusion matrix was also generated to see the summary of the predictions made.



```
## Predicting the results using the test set
heart_pred <- predict(heart_tree2, heart_data[-train,], type="class")
heart_pred

## Creating a confusion matrix
cm <- with(heart_data[-train,], table(heart_pred, target))
cm
```

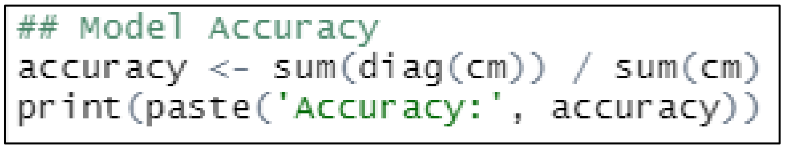


```
> ## Predicting the results using the test set
> heart_pred <- predict(heart_tree2, heart_data[-train,], type="class")
> heart_pred
  [1] 0 0 1 0 0 0 1 0 1 1 0 0 1 1 1 1 0 0 0 0 0 1 0 0 0 1 0 1 0 1 0 1 0 1 1 1 1 0 1 1 0 0 0 0 0 1 1 1 1
 [50] 1 1 1 1 1 1 1 1 0 1 1 1 0 0 1 1 1 0 1 1 1 0 1 0 1 0 0 0 1 0 0 1 1 0 1 1 0 0 0 0 1 0 1 0 1 1 1 0 1
 [99] 1 1 0 0 1 0 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 0 1 0 1 1 1 0 1 1 1 0 0 1 0 1 0 0 1 0 0 1 0 0 0 0 1 1 1
[148] 0 0 0 1 1 0 0 1 1 0 1 0 1 1 1 0 0 1 0 1 1 1 1 1 0 1 0 0 0 1 0 1 1 1 0 0 1 0 1 1 1 1 0 1 1 1 1 1 0
[197] 0 1 0 1 1 1 1 1 1 0 1 1 0 0 0 0 1 1 1 0 1 1 1 0 0 1 1 0 0 1 1 0 1 1 0 0 1 1 0 1 0 1 1 0 1 1 0 0 0
[246] 0 0 1 0 1 1 1 1 0 1 1 0 0 1 0 1 1 0 1 1 1 0 1 1 1 1 0 1 0 0 1 1 1 0 0 0 0 0 1 0 1 0 1 1 1 0 0 1 1
[295] 1 1 0 0 0 1 0 1 1 1 1 1 1 1 1 1 0 0 1 1 1 0 0 1 0 0 0 1 1 0 0 0 0 0 0 1 1 1 1 0 1 1 1 1 1 1 0 1 1
[344] 0 0 0 1 1 1 0 1 0 1 0 0 1 0 1 1 0
Levels: 0 1
```

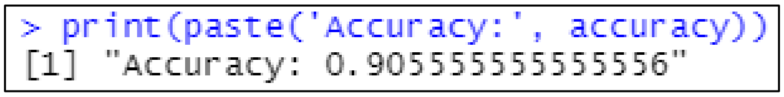


```
> cm <- with(heart_data[-train,], table(heart_pred, target))
> cm
          target
heart_pred   0   1
         0 143  10
         1  24 183
```

To evaluate the performance of the model in terms of prediction, the accuracy was calculated. To get the accuracy percentage, the number of correct predictions is divided by the total number of predictions made. Based on the obtained value of accuracy, the decision tree model built performs well as it was able to get an accuracy of 90.56%.

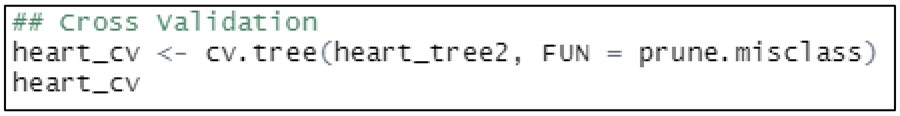


```
## Model Accuracy
accuracy <- sum(diag(cm)) / sum(cm)
print(paste('Accuracy:', accuracy))
```

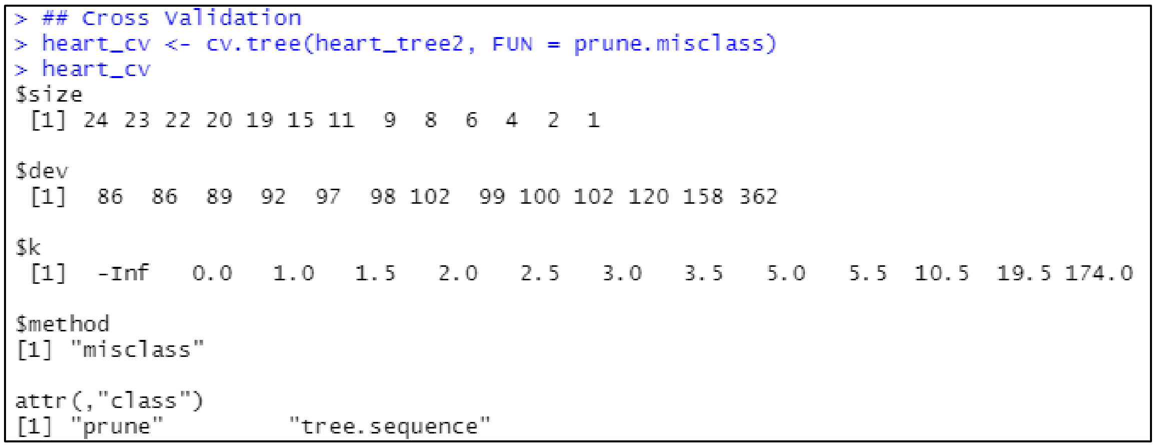


```
> print(paste('Accuracy:', accuracy))
[1] "Accuracy: 0.905555555555556"
```

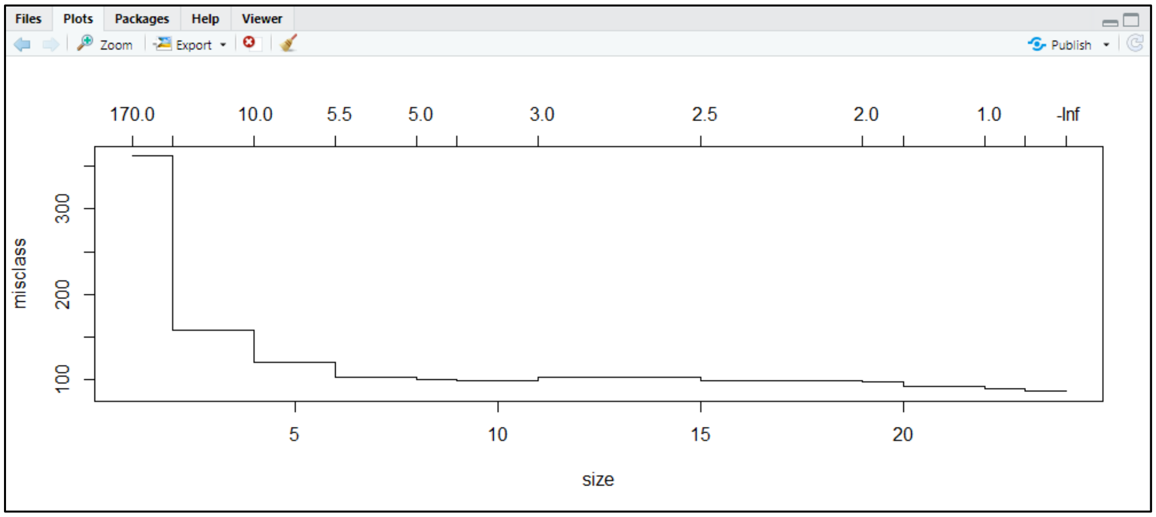
The model then underwent a cross-validation to prune the tree more effectively and to verify how accurate the model is on multiple and different subsets of data. The cross validation model was also visualized through the plot() function.



```
## Cross Validation
heart_cv <- cv.tree(heart_tree2, FUN = prune.misclass)
heart_cv
```

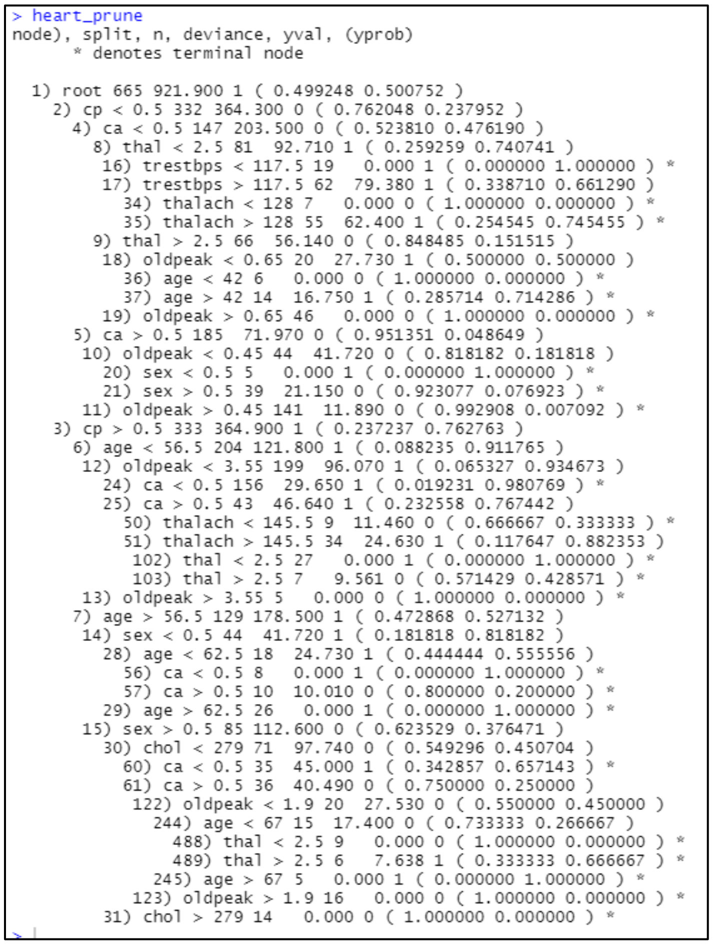


```
> ## Cross Validation
> heart_cv <- cv.tree(heart_tree2, FUN = prune.misclass)
> heart_cv
$size
 [1] 24 23 22 20 19 15 11  9  8  6  4  2  1

$dev
 [1]  86  86  89  92  97  98 102  99 100 102 120 158 362

$k
 [1]  -Inf   0.0   1.0   1.5   2.0   2.5   3.0   3.5   5.0   5.5  10.5  19.5 174.0

$method
[1] "misclass"

attr(,"class")
[1] "prune"         "tree.sequence"
```



After cross-validation, the model was pruned using the prune.misclass() function wherein the size of the three (number of terminal nodes) is specified.

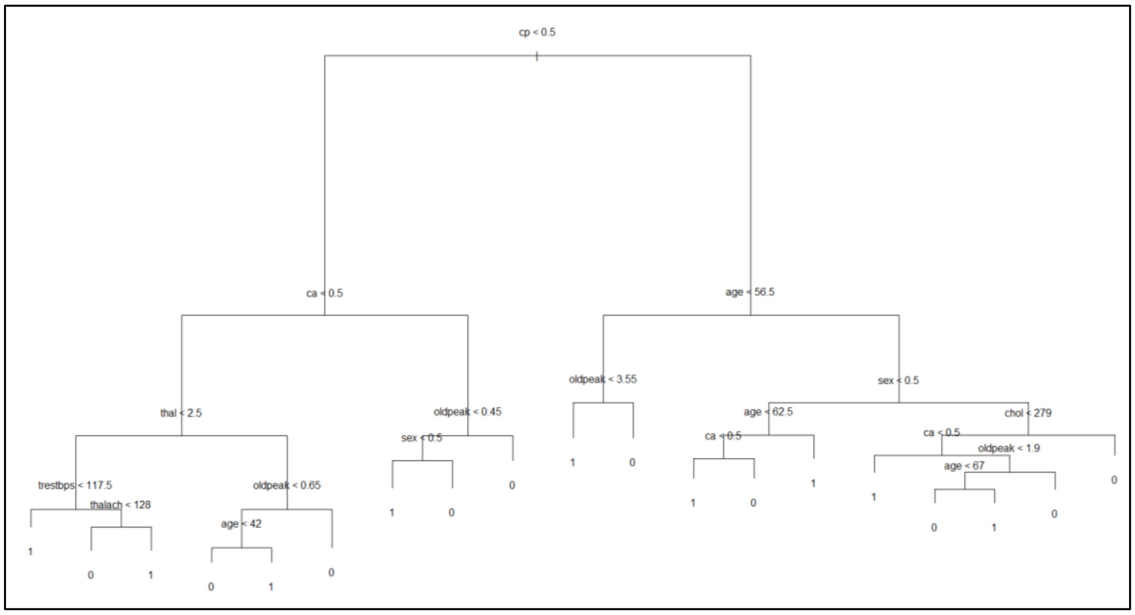
```r
## Pruning model
heart_prune <- prune.misclass(heart_tree2, best = 23)
heart_prune
```



```
> heart_prune
node), split, n, deviance, yval, (yprob)
      * denotes terminal node

  1) root 665 921.900 1 ( 0.499248 0.500752 )
    2) cp < 0.5 332 364.300 0 ( 0.762048 0.237952 )
      4) ca < 0.5 147 203.500 0 ( 0.523810 0.476190 )
        8) thal < 2.5 81  92.710 1 ( 0.259259 0.740741 )
         16) trestbps < 117.5 19   0.000 1 ( 0.000000 1.000000 ) *
         17) trestbps > 117.5 62  79.380 1 ( 0.338710 0.661290 )
           34) thalach < 128 7   0.000 0 ( 1.000000 0.000000 ) *
           35) thalach > 128 55  62.400 1 ( 0.254545 0.745455 ) *
        9) thal > 2.5 66  56.140 0 ( 0.848485 0.151515 )
         18) oldpeak < 0.65 20  27.730 1 ( 0.500000 0.500000 )
           36) age < 42 6   0.000 0 ( 1.000000 0.000000 ) *
           37) age > 42 14  16.750 1 ( 0.285714 0.714286 ) *
         19) oldpeak > 0.65 46   0.000 0 ( 1.000000 0.000000 ) *
      5) ca > 0.5 185  71.970 0 ( 0.951351 0.048649 )
       10) oldpeak < 0.45 44  41.720 0 ( 0.818182 0.181818 )
         20) sex < 0.5 5   0.000 1 ( 0.000000 1.000000 ) *
         21) sex > 0.5 39  21.150 0 ( 0.923077 0.076923 ) *
       11) oldpeak > 0.45 141  11.890 0 ( 0.992908 0.007092 ) *
    3) cp > 0.5 333 364.900 1 ( 0.237237 0.762763 )
      6) age < 56.5 204 121.800 1 ( 0.088235 0.911765 )
       12) oldpeak < 3.55 199  96.070 1 ( 0.065327 0.934673 )
         24) ca < 0.5 156  29.650 1 ( 0.019231 0.980769 ) *
         25) ca > 0.5 43  46.640 1 ( 0.232558 0.767442 )
           50) thalach < 145.5 9  11.460 0 ( 0.666667 0.333333 ) *
           51) thalach > 145.5 34  24.630 1 ( 0.117647 0.882353 )
            102) thal < 2.5 27   0.000 1 ( 0.000000 1.000000 ) *
            103) thal > 2.5 7   9.561 0 ( 0.571429 0.428571 ) *
       13) oldpeak > 3.55 5   0.000 0 ( 1.000000 0.000000 ) *
      7) age > 56.5 129 178.500 1 ( 0.472868 0.527132 )
       14) sex < 0.5 44  41.720 1 ( 0.181818 0.818182 )
         28) age < 62.5 18  24.730 1 ( 0.444444 0.555556 )
           56) ca < 0.5 8   0.000 1 ( 0.000000 1.000000 ) *
           57) ca > 0.5 10  10.010 0 ( 0.800000 0.200000 ) *
         29) age > 62.5 26   0.000 1 ( 0.000000 1.000000 ) *
       15) sex > 0.5 85 112.600 0 ( 0.623529 0.376471 )
         30) chol < 279 71  97.740 0 ( 0.549296 0.450704 )
           60) ca < 0.5 35  45.000 1 ( 0.342857 0.657143 ) *
           61) ca > 0.5 36  40.490 0 ( 0.750000 0.250000 )
            122) oldpeak < 1.9 20  27.530 0 ( 0.550000 0.450000 )
              244) age < 67 15  17.400 0 ( 0.733333 0.266667 )
                488) thal < 2.5 9   0.000 0 ( 1.000000 0.000000 ) *
                489) thal > 2.5 6   7.638 1 ( 0.333333 0.666667 ) *
              245) age > 67 5   0.000 1 ( 0.000000 1.000000 ) *
            123) oldpeak > 1.9 16   0.000 0 ( 1.000000 0.000000 ) *
         31) chol > 279 14   0.000 0 ( 1.000000 0.000000 ) *
>
```

The pruned model/tree was visualized using the plot() function. After cross-validation and pruning, the terminal nodes of the tree were reduced, making the tree look simpler and easily understandable.

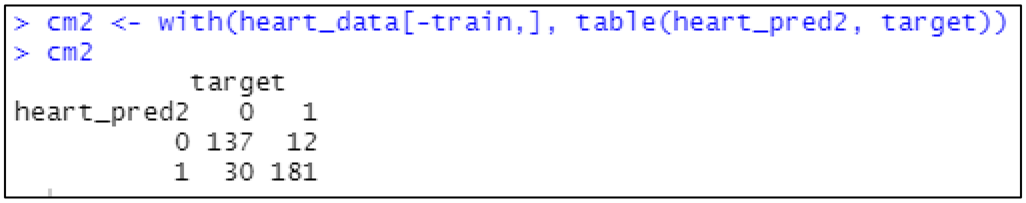
```
# Plotting the Pruned Tree
plot(heart_prune)

text(heart_prune, pretty=0)
```



After conducting cross-validation and pruning, the decision tree model did another prediction to check if there are changes, whether improvement or decline, on the performance of the model in terms of accuracy. Just like the previous prediction, a confusion matrix was also generated here to see the summary of predictions.

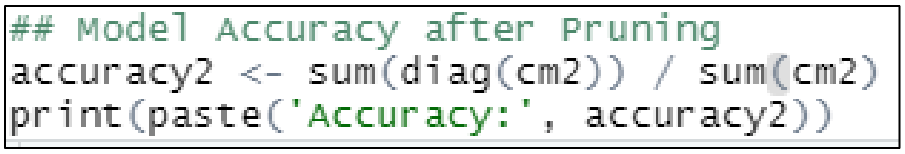
```
## Predicting the results AGAIN using the test set AFTER pruning
heart_pred2 <- predict(heart_prune, heart_data[-train,], type="class")
heart_pred2

cm2 <- with(heart_data[-train,], table(heart_pred2, target))
cm2
```

```
> heart_pred2
  [1] 0 0 1 0 0 0 1 0 1 1 1 0 1 1 1 1 0 0 0 0 0 0 0 0 1 1 0 1 0 1 0 1 1 1 1 1 1 0 0 1 0 0 0 0 0 1 1 1
 [49] 1 1 1 1 1 1 1 1 1 0 1 1 1 0 0 1 1 1 0 1 1 1 0 1 0 1 0 0 0 1 0 0 1 1 0 1 1 0 0 0 1 1 0 1 1 1 1 1
 [97] 0 1 1 1 0 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 0 1 1 0 0 1 1 1 0 0 1 0 1 0 0 1 0 1 1 0 1 0 0
[145] 1 1 1 0 0 0 1 1 0 0 1 1 0 1 0 1 1 1 0 0 1 0 1 1 1 1 1 0 0 0 0 0 1 0 1 1 1 0 0 1 1 1 1 1 1 0 1 1
[193] 1 1 1 0 0 1 0 1 1 1 1 1 1 0 1 1 0 0 0 0 1 1 1 0 1 1 1 0 0 1 1 0 0 1 1 0 1 1 0 0 1 1 0 1 0 1 1 0
[241] 1 1 0 0 0 0 0 1 0 1 1 1 1 0 1 1 1 0 1 0 1 1 0 1 0 1 0 1 0 1 1 0 1 0 0 1 1 1 0 0 0 0 0 1 0 1 1 1
[289] 1 1 0 0 1 0 1 1 0 0 0 1 0 1 1 1 1 1 1 1 1 1 0 0 1 1 1 0 0 1 0 0 0 1 1 0 0 0 0 1 0 1 1 1 1 1 0 1 1
[337] 1 1 0 1 0 1 1 0 0 0 1 1 1 0 1 0 1 0 0 1 0 1 1 0
Levels: 0 1
```

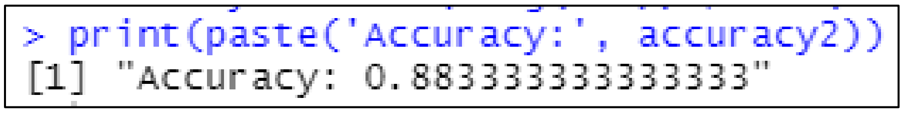


```
> cm2 <- with(heart_data[-train,], table(heart_pred2, target))
> cm2
           target
heart_pred2   0   1
          0 137  12
          1  30 181
```

Another performance evaluation was conducted but this time using the model that have been cross-validated and pruned, which also underwent another prediction stage.



```
## Model Accuracy after Pruning
accuracy2 <- sum(diag(cm2)) / sum(cm2)
print(paste('Accuracy:', accuracy2))
```



```
> print(paste('Accuracy:', accuracy2))
[1] "Accuracy: 0.883333333333333"
```

Based on the yielded result, the new model (after cross validation and pruning) has an accuracy percentage of 88.33%, which is lower than the accuracy value of the first prediction when there were no cross validation and pruning yet. However, it can still be concluded that the model has a good performance in predicting the class labels for being able to obtain an 88.33% accuracy value. In addition, getting a lower accuracy after cross validation and pruning means that for this particular dataset and algorithm, those processes are not that necessary.

VI. CONCLUSION

After implementing the Decision tree algorithm using the Heart Disease Prediction dataset, the following conclusions are made:

- The Heart Disease Prediction dataset was checked to look for any missing values that may affect the model creation and prediction process. No missing values were found in the dataset; thus, data cleaning was not conducted. In addition, no other formats in the dataset were found to be a hindrance in successfully fitting the model except for the target variable which was at first set to integer format. Hence, as.factor() function was used to convert the target variable from integer to factor format to be able to build the model.
- When the decision tree model was built and prior to the split of training and testing sets, not all variables are selected in the tree construction such as cp, ca, thal, exang, age, restecg, slope, oldpeak, chol, sex, and thalach. The number of terminal nodes is 29, the residual mean deviance is 0.265, while the misclassification error rate is 0.05756. However, a different set of results were produced with the model after the dataset was split into training (65%) and

testing sets (35%). The selected variables are reduced having only the following: cp, ca, thal, trestbps, thalach, oldpeak, age, sex and chol. The residual mean deviance raised to 0.3342 and the misclassification error rate also raised from 0.05756 to 0.07068.

- The model did two sets of predictions. The first one was after the model was built and then the dataset was split into training and testing sets, and the second was after cross-validation and pruning. The first prediction was able to produce an accuracy rate of 90.56%, while the second prediction had a small amount of decrease to 88.33%. However, it can still be concluded that the model was able to perform well in prediction with or without pruning based on the accuracy values it had. For this particular algorithm and dataset, cross-validation and pruning may not be necessary anymore as there was a decrease in accuracy that has been seen. However, those processes can still be considered if they want to reduce the terminal nodes and to make the tree look simpler and more understandable.

VII. REFERENCES

[1] "Medical Disease Prediction using Machine Learning Algorithms," Ijraset.com, 2013. https://www.ijraset.com/research-paper/medical-disease-prediction-using-ml-algorithms#:~:text=Large%20amounts%20of%20data%20such,and%20expertise%20of%20an%20individual. (accessed Jul. 28, 2022).

[2] Aunalytics, "Decision Trees: An Overview - Aunalytics," Aunalytics, Jan. 31, 2015. https://www.aunalytics.com/decision-trees-an-overview/#:~:text=Decision%20trees%20tend%20to%20be,are%20two%20stages%20to%20prediction. (accessed Jul. 28, 2022).

[3] "Decision Tree Analysis: Choosing by Projecting 'Expected Outcomes,'" Mindtools.com, 2017. https://www.mindtools.com/dectree.html#:~:text=Decision%20trees%20provide%20an%20effective,the%20probabilities%20of%20achieving%20them. (accessed Jul. 28, 2022).

[4] P. Saha, "Heart Disease Prediction," Kaggle.com, 2020. https://www.kaggle.com/datasets/puspitasaha/heart-disease-prediction (accessed Jul. 28, 2022).