**Analysis of Heart Dataset**

**Introduction:**

This dataset contains 270 observations and 14 attributes as below.

1. age
2. sex
3. chest pain type (4 values)
4. resting blood pressure
5. serum cholestoral in mg/dl
6. fasting blood sugar > 120 mg/dl
7. resting electrocardiographic results (values 0,1,2)
8. maximum heart rate achieved
9. exercise induced angina
10. oldpeak = ST depression induced by exercise relative to rest
11. the slope of the peak exercise ST segment
12. number of major vessels (0-3) colored by flourosopy
13. thal: 3 = normal; 6 = fixed defect; 7 = reversable defect
14. heart prediction: 1 = Absence; 2 = Presence of Heart Disease

The last column represents the response, while the rest are the predictors.

The columns 1,4,5,8,10,12 hold the numerical/continuous data.

The data in the 11th column is ordered, whereas it is binary in columns 2,6,9.

Columns 7,3,13,14 hold the nominal data which is categorical by default.

The main objective is to build models using classification trees and random forests, and to find the most accurate model among them.

Following are the steps to be followed in R:

1. **Exploratory Data Analysis (EDA).**

Exploratory Data Analysis (EDA) is the first step before model building. It is done by cleaning the data and for detecting any pattern(s) or irregularities in the dataset.

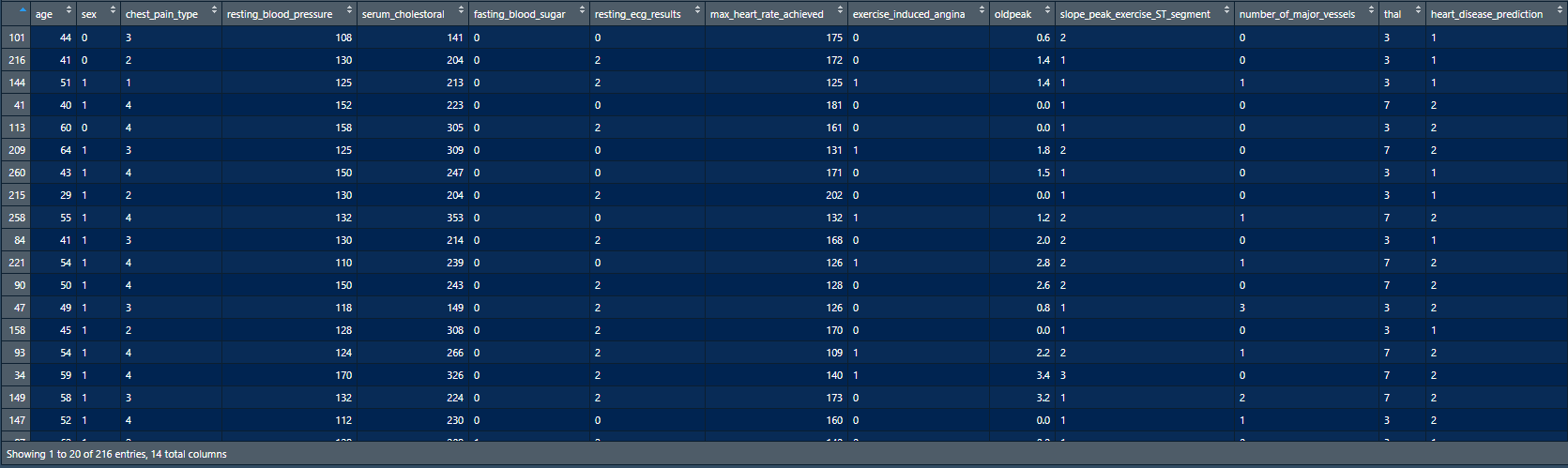
The below document covers the EDA.

****

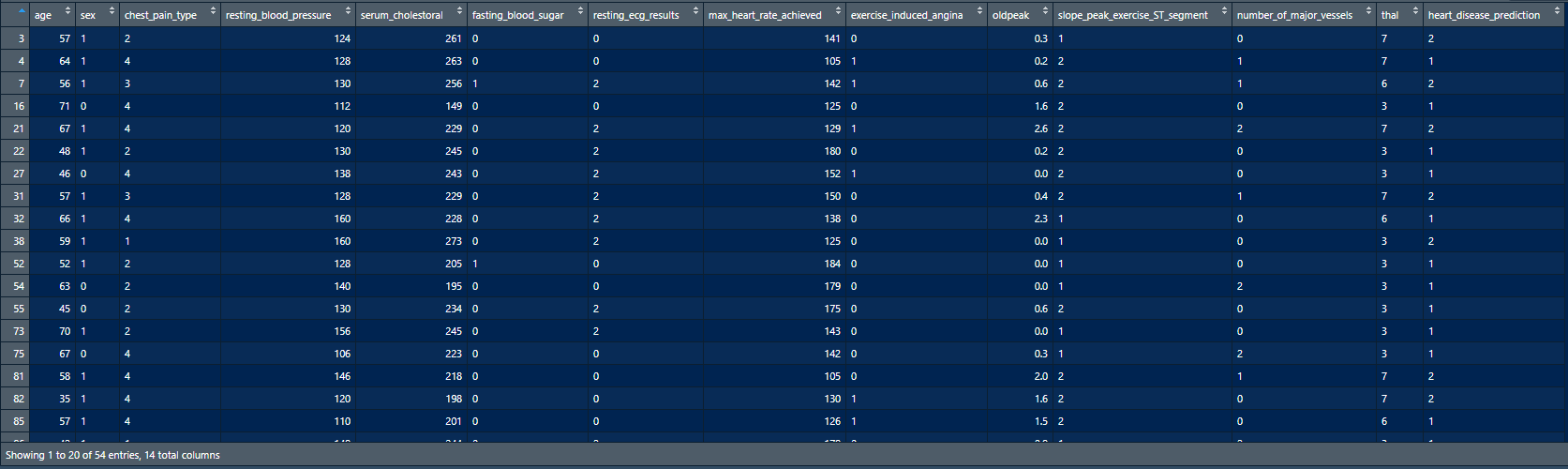
1. **Splitting the dataset into 80% training and 20% test set.**

Here, seeding must be done to control the randomness while splitting. Since there are 270 observations in the dataset, 216 observations would be the training data and the rest 54, the testing data.

**Training data:**



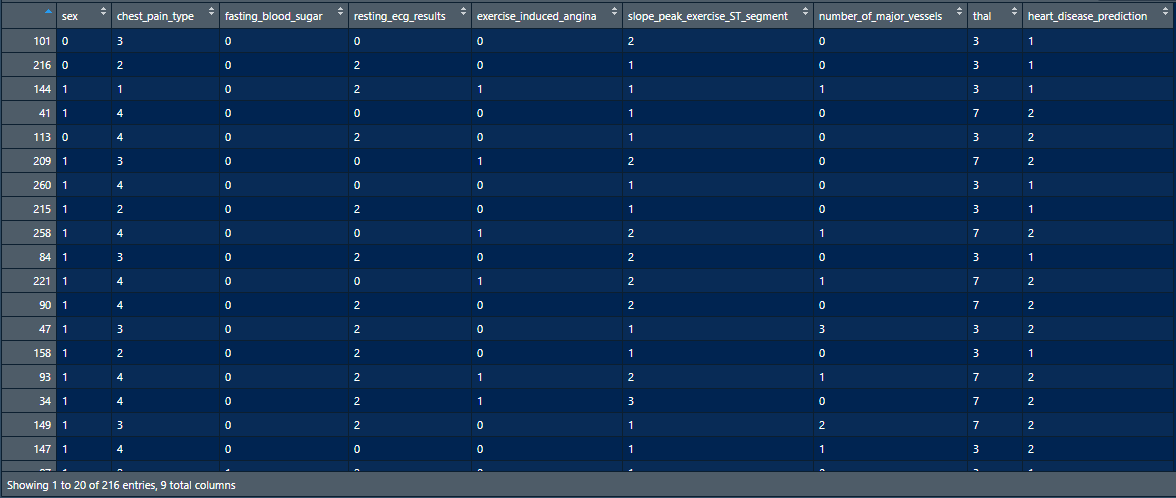
**Testing data:**



1. **Finding the root node of the training data using only the categorical type predictor variables. Here, the number of splits at the root node can equal the number of categories in the predictor variable being split.**

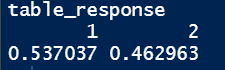
The root node can be found using the entropy and information gain of the predictors with respect to the response variable. It is better to create a training data set that holds the categorical data only.

**Training data with categorical data only:**



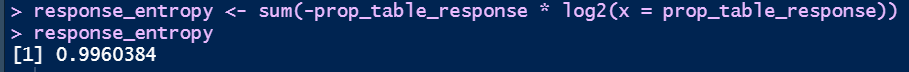
With this data set, entropy and information gain needs to be calculated for categorical type predictors. First, the entropy of the response variable needs to be calculated. This can be done using the proportion table, as it provides the probability information that is used in calculating entropy. "heart\_disease\_prediction" is the response variable and the last column in the data frame. So, a frequency table for the response variable is created, which is then used for creating the proportion table.

**Frequency table:**



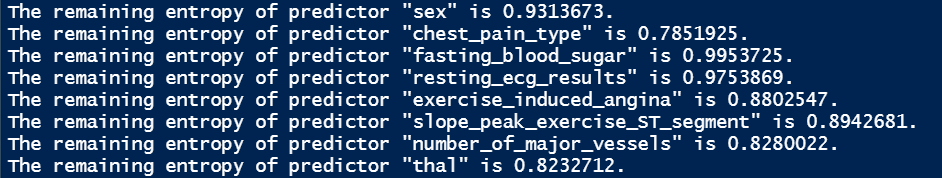
The entropy is calculated from the proportion table using Shannon's formula.

**Entropy of the response variable:**

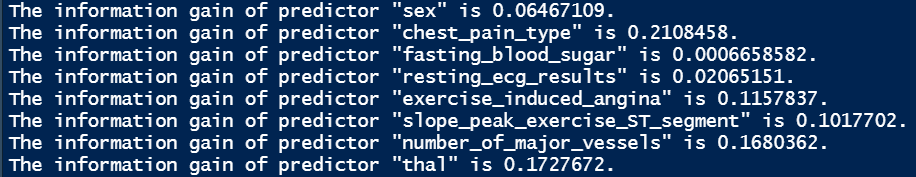


Now the remaining entropy of each predictor w.r.t response variable needs to be calculated. Information gain can then be calculated by subtracting remaining entropy from response/overall entropy.

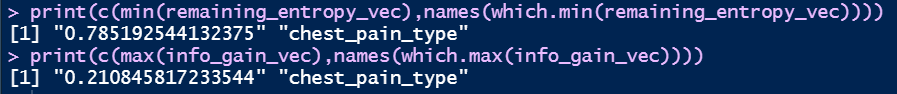
**Remaining entropy of the predictors:**



**Information gain of the predictors:**

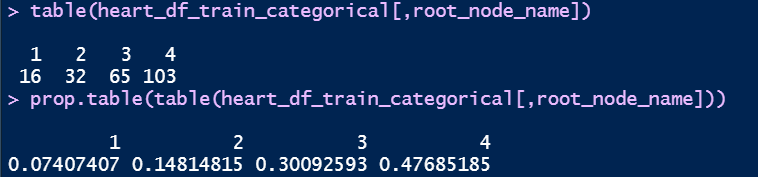


**Predictor with lowest remaining entropy/highest information gain:**



The "chest\_pain\_type" predictor has the highest information gain when compared to the other predictors. This is because it has the least remaining entropy. Hence, the "chest\_pain\_type" predictor is to be used for the root node split. If other predictors are used, then the tree may become complex and more diverse. It will be hard to predict with this model, as there would be more errors and high bias. There is high chance for misclassification. The "chest\_pain\_type" predictor contains only 4 values. So, the root node split will occur based on these values. Thus, the number of splits is equal to 4.

**Frequency and proportion tables of the root node column:**

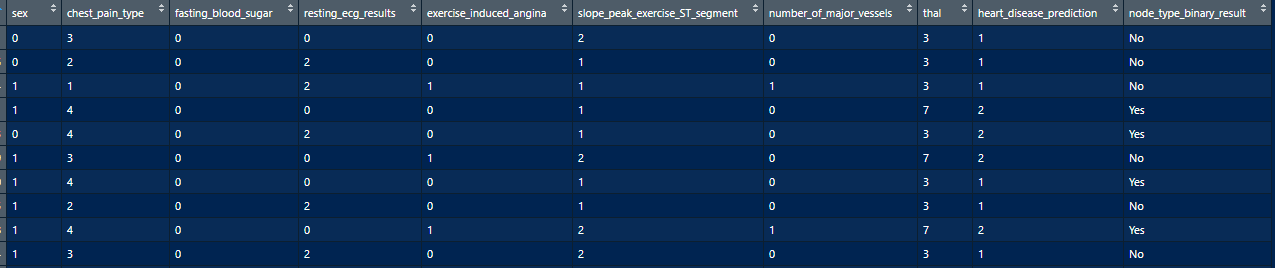


The highest percentage of people suffer from chest pain Type "4", while the least suffer from chest pain Type "1". Hence it would be easier to predict if people are suffering from chest pain Type "4" than chest pain Type "1". However, people suffering from Type "1" have the lowest number of observations. This means the tree becomes less complex and diverse as it goes down.

1. **Redoing (3.), except there can only be a binary split i.e. a split with only 2 possible outcomes.**

Since the "chest\_pain\_type" predictor contains 4 values, a binary split must be done with the value that has the highest percentage or probability. One outcome will hold "Yes" if type matches the one with max probability. Else it will be "No". Here, the outcome with the highest percentage or probability is “4”.

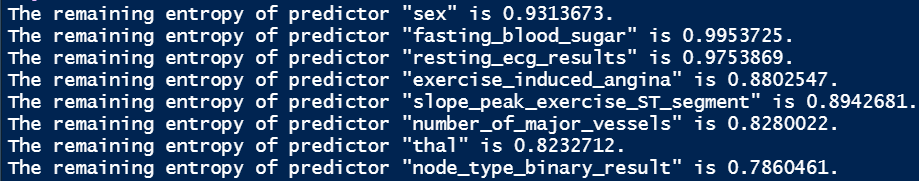
**The transformed data of the "chest\_pain\_type" predictor:**



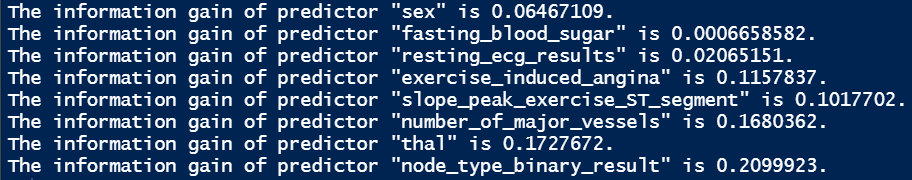
The “node\_type\_binary\_result” is the transformed version of the “chest\_pain\_type" predictor.

Now, the remaining entropy of each predictor w.r.t response variable needs to be re-calculated considering the new column. The response column and "chest\_pain\_type" predictor need not be included for calculation.

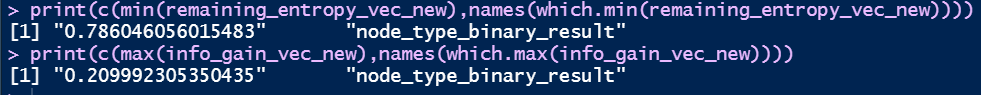
**Remaining entropy of the predictors:**



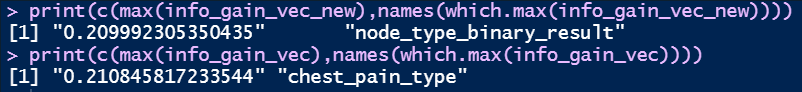
**Information gain of the predictors:**



**Predictor with lowest remaining entropy/highest information gain:**



**Information gain of the transformed and original "chest\_pain\_type" predictor:**



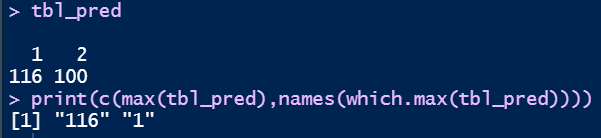
Here, the new predictor "node\_type\_binary\_result" has the highest information gain when compared to the other predictors. Plus, the information gain of "node\_type\_binary\_result" predictor is almost same as that of the "chest\_pain\_type" predictor. This means similar information can be obtained from predictors "node\_type\_binary\_result" or "chest\_pain\_type".

This implies that a binary split could be used on the "node\_type\_binary\_result" predictor. Binary split would simplify the tree structure and allow predictions with higher accuracy. There would be less chance for misclassification when are only two outcomes. When there are 4 outcomes, the chance for misclassification is bit higher. The prediction for the other levels in "chest\_pain\_type" predictor via binary split on the "node\_type\_binary\_result" predictor is less complicated and could be done without much hurdle.

1. **Redoing (4.), but with the inclusion of the continuous numeric predictor variables possible outcomes. A binary split must be done here as well.**

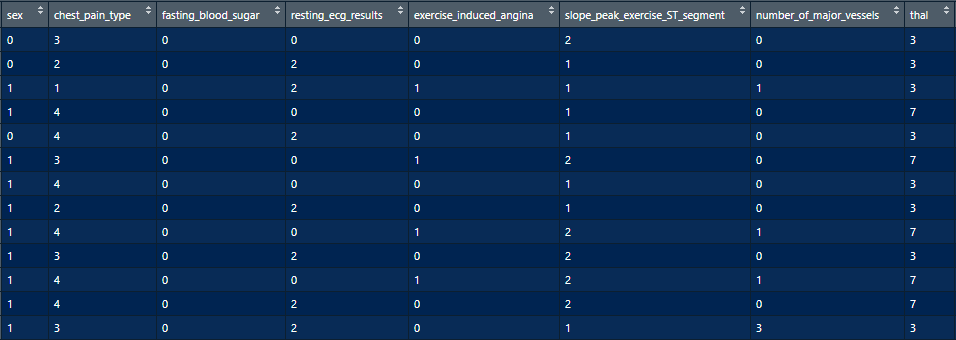
The original training data can be used here, as it already contains the numeric columns. The numerical data must be transformed into categorical data by splitting it into two. This can be done by finding the response category with the highest count and taking that data alone. The split will be done on the individual numeric columns on their corresponding mean in that data. A frequency table with the response variable is created to identify which category has the highest count.

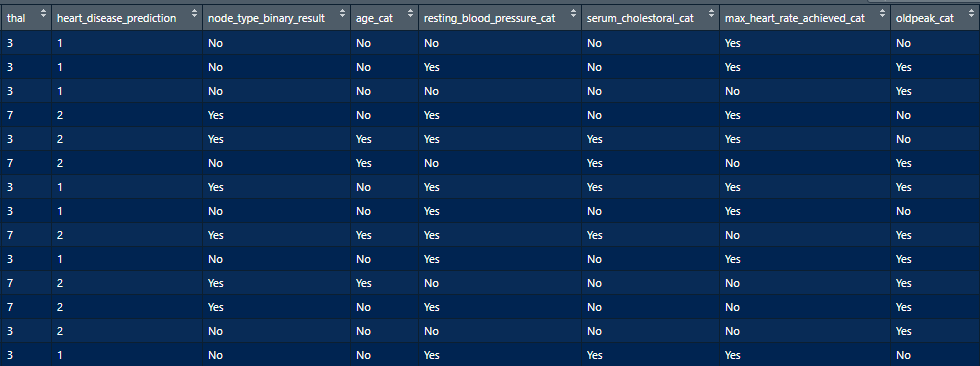
**Frequency table of the response column:**



Here, response category "1" has the highest count of 116. Since the split in numerical data can lead to creation of many columns, the training data will be backed up and the numerical data will be replaced by its corresponding categorical data. If the value of the numeric predictor is greater than its mean, then “Yes” would be stored in its respective categorical column. Else, “No” would be stored. The predictor with the “cat” suffix denotes the new predictor.

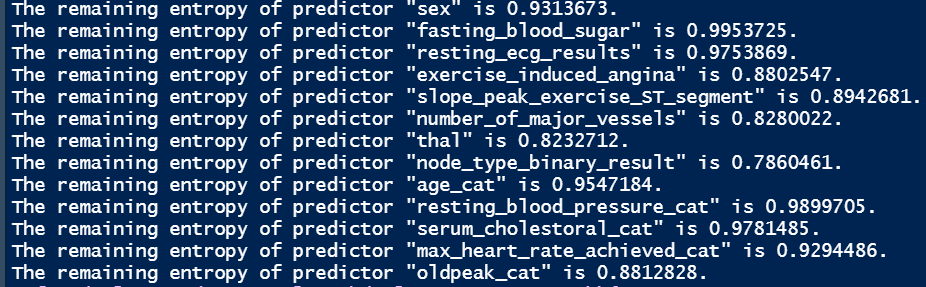
**Training data with numeric to categorical data transformation:**



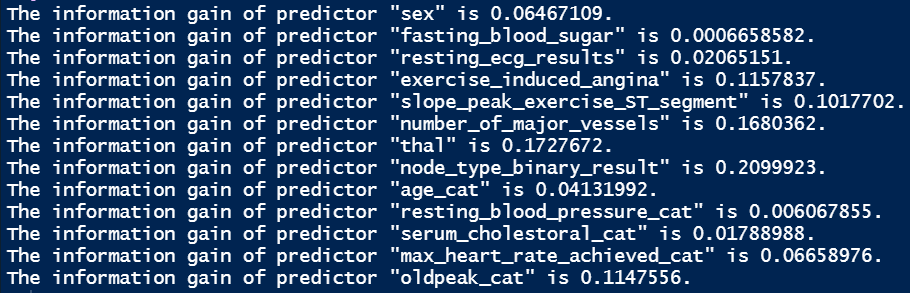


Now, the remaining entropy of each predictor w.r.t response variable needs to be re-calculated considering the new columns. The response column and "chest\_pain\_type" predictor need not be included for calculation.

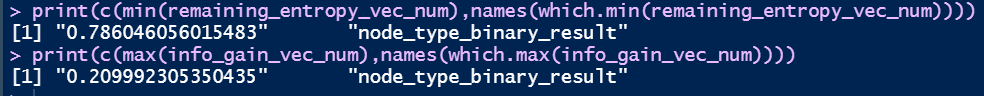
**Remaining entropy of the predictors:**



**Information gain of the predictors:**

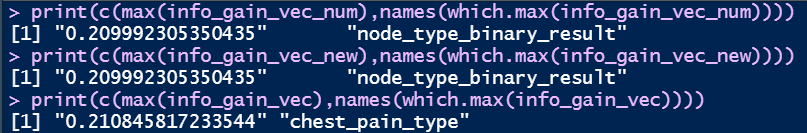


**Predictor with lowest remaining entropy/highest information gain:**



Even after the inclusion of the continuous numeric predictors, the information gain of the predictor "node\_type\_binary\_result" is the highest among all predictors.

**Information gain of the transformed predictor (after and before numeric predictor inclusion) and original "chest\_pain\_type" predictor:**



This implies that a binary split could be used on the "node\_type\_binary\_result" predictor as it would simplify the tree structure and allow predictions with higher accuracy. There would be less chance for misclassification with only two outcomes. Hence, the "node\_type\_binary\_result" predictor can be considered as the root node. The "node\_type\_binary\_result" predictor is the binary transformed data of "chest\_pain\_type" predictor in the original training data.

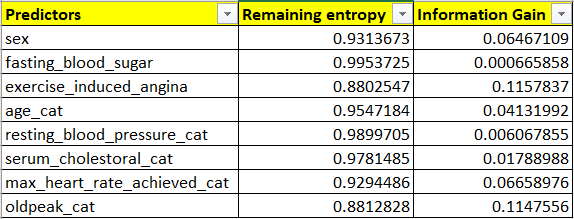
1. **Investigating the next level of split, i.e. finding the predictor variable(s) to split the first split found in part (5.). Only binary splits are allowed again here.**

The "node\_type\_binary\_result" predictor is the root node element. It contains two values "Yes" and "No". As there are two values, the entropy and information gain must be calculated for the training data corresponding to those values. Which means that the entropy and information gain will be calculated twice. One for training data with the "Yes" value in "node\_type\_binary\_result" predictor and the other with the "No" value.

**In the training data, the below columns hold 2 types of data (binary results):**

"sex", "fasting\_blood\_sugar", "exercise\_induced\_angina", "age\_cat", "resting\_blood\_pressure\_cat", "serum\_cholestoral\_cat", "max\_heart\_rate\_achieved\_cat", "oldpeak\_cat"

**Below are the remaining entropy and information gain values:**

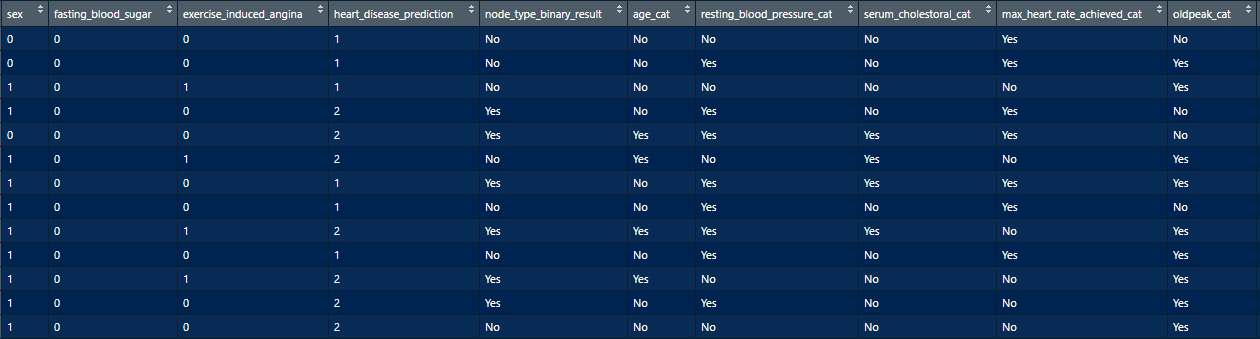


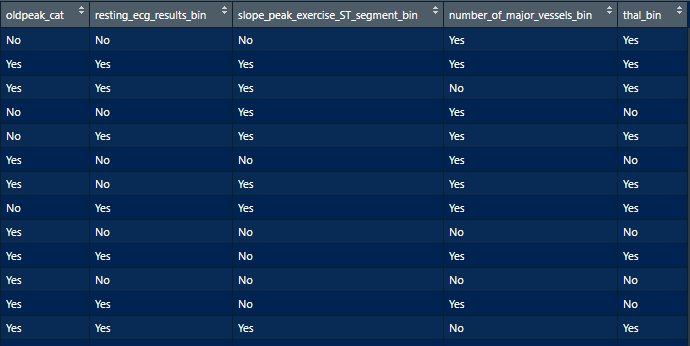
Since a binary split must be done, the categorical data that hold more than 2 levels, also need to be transformed to hold 2 levels. The approach is like (4.) and should be done for all the columns that hold more than 2 levels. On this dataset, the remaining entropy and information gain should be carried out to find the terminal nodes. This would be the next level of split with binary outcomes.

1. **Analysis of the results from the code used for implementing (6.).**

The categorical data that hold more than 2 needs to be transformed to hold 2 levels (like (4.)). As there could be creation of many columns, training data is backed up. "chest\_pain\_type" category could be removed from the dataset. The mode of the predictor is taken and compared with the data to get the categorical value. The original column is to be removed and instead it will hold the binary categorical value. If the predictor value is equal to its mode (most occurred value), then “Yes” is stored as the binary categorical value. Else, it is “No”. The predictor with the “bin” suffix denotes the new predictor.

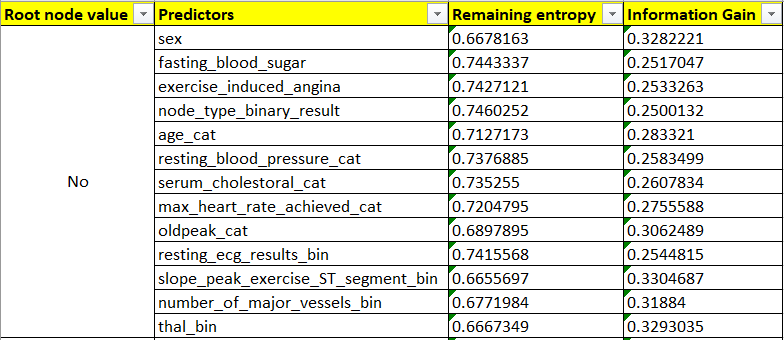
**Training data with categorical to binary type categorical transformation:**



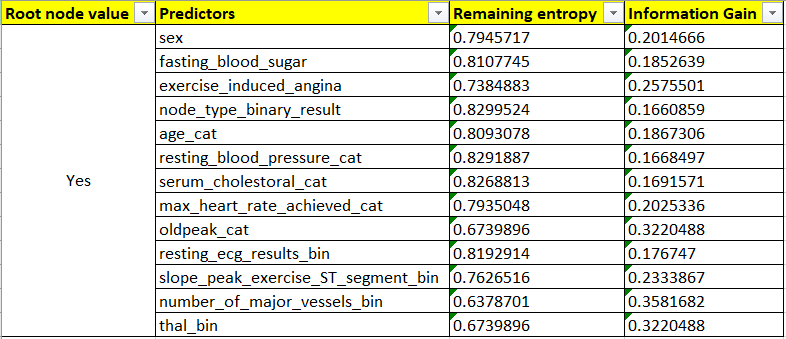


For every value in the root node, the Remaining Entropy and information gain must be calculated with respect to the training data. The response variable need not be included for calculation.

**Remaining entropy and information gain of the predictors with node level value as “No”:**



**Remaining entropy and information gain of the predictors with node level value as “Yes”:**



**Predictor with lowest remaining entropy/highest information gain for node level value as “No”:**



**Predictor with lowest remaining entropy/highest information gain for node level value as “Yes”:**



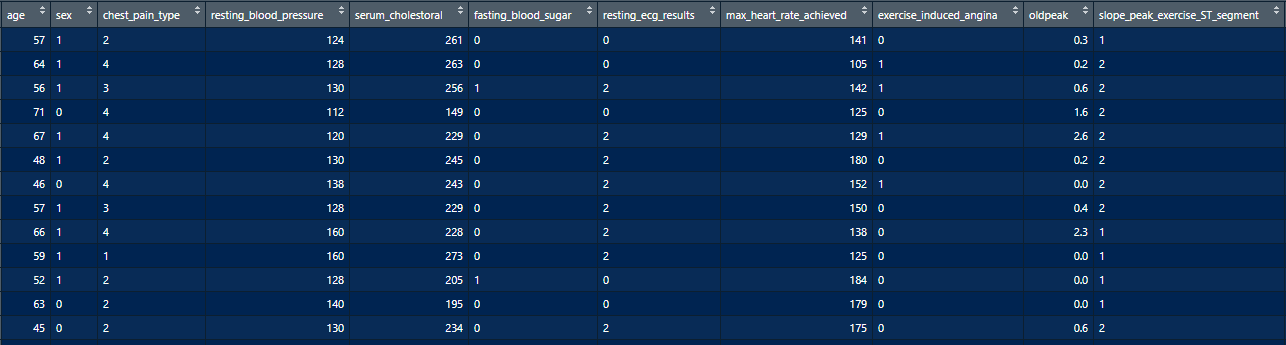
For the "No" value in "node\_type\_binary\_result" predictor, the terminal node would be the binary predictor "slope\_peak\_exercise\_ST\_segment\_bin", which is the transformed version of the "slope\_peak\_exercise\_ST\_segment" predictor. For the "Yes" value in "node\_type\_binary\_result" predictor, the terminal node would be the binary predictor "number\_of\_major\_vessels\_bin", which is the transformed version of the "number\_of\_major\_vessels" predictor.

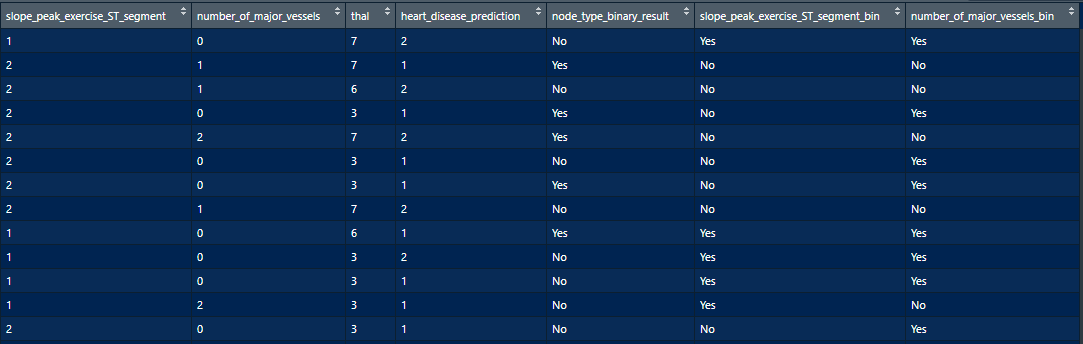
1. **Prediction of the class for the test dataset and creation of confusion matrix.**

To predict the class for the test data set, the root node and terminal node columns present in the test data set needs to be transformed to hold binary values as done before for the training datasets.

The predictor with the “bin” suffix denotes the new predictor.

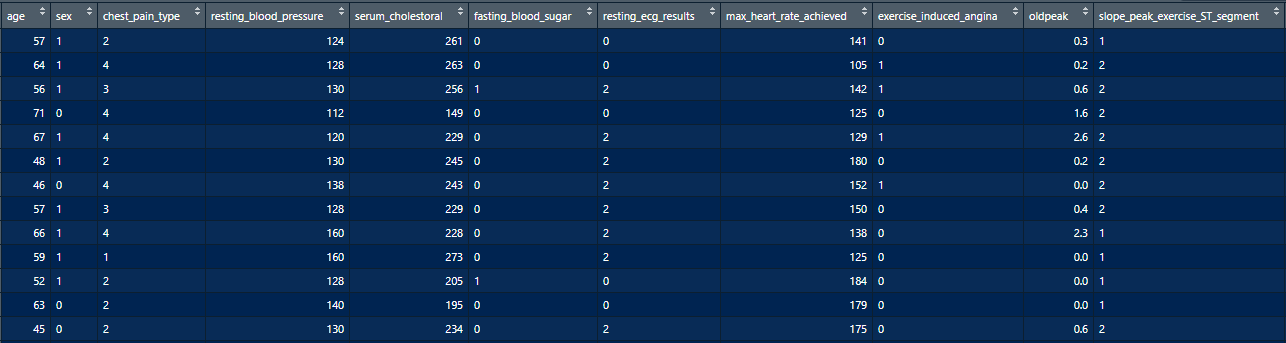
**Testing data with categorical to binary type categorical transformation:**

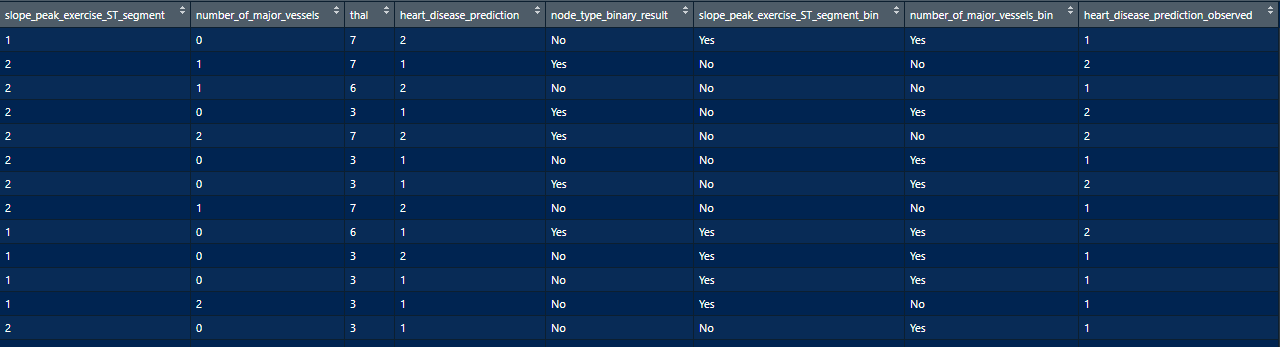




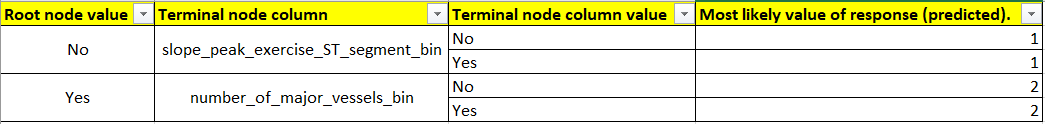
Once the root and terminal nodes are created, those values should be mapped to the training data set. For each row, the root and terminal node value is mapped with the training data, and then the response value with the highest probability/count will be taken. This would be considered as the predicted value.

**Testing data with predicted response value:**

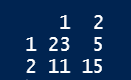




**Predicted values of the response based on the root and terminal node values:**

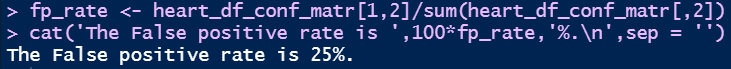


**Confusion Matrix:**



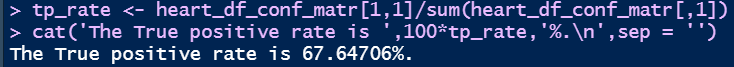
The rows represent the predicted values, while the columns represent the actual values.

False positive rate is the number of false positive entries divided by the sum of number of false positive and true negative entries.



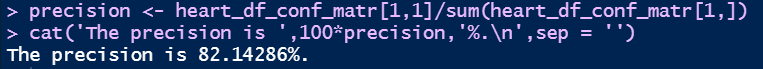
The False positive rate is 25%.

True positive rate is the number of true positive entries divided by the sum of number of true positive and false negative entries.



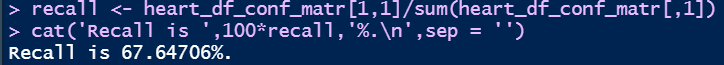
The True positive rate is 67.64%.

Precision is the number of true positive entries divided by the sum of number of true positive and false positive entries.



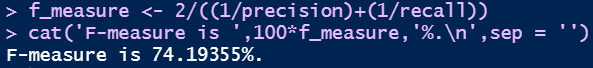
The precision is 82.14%.

Recall is the number of true positive entries divided by the sum of number of true positive and false negative entries.



The recall is 67.64%.

F-measure is 2 divided by the sum of reciprocals of precision and recall values.



F-measure is 74.19355%.

Baseline rate is the probability of the response value with the highest count.



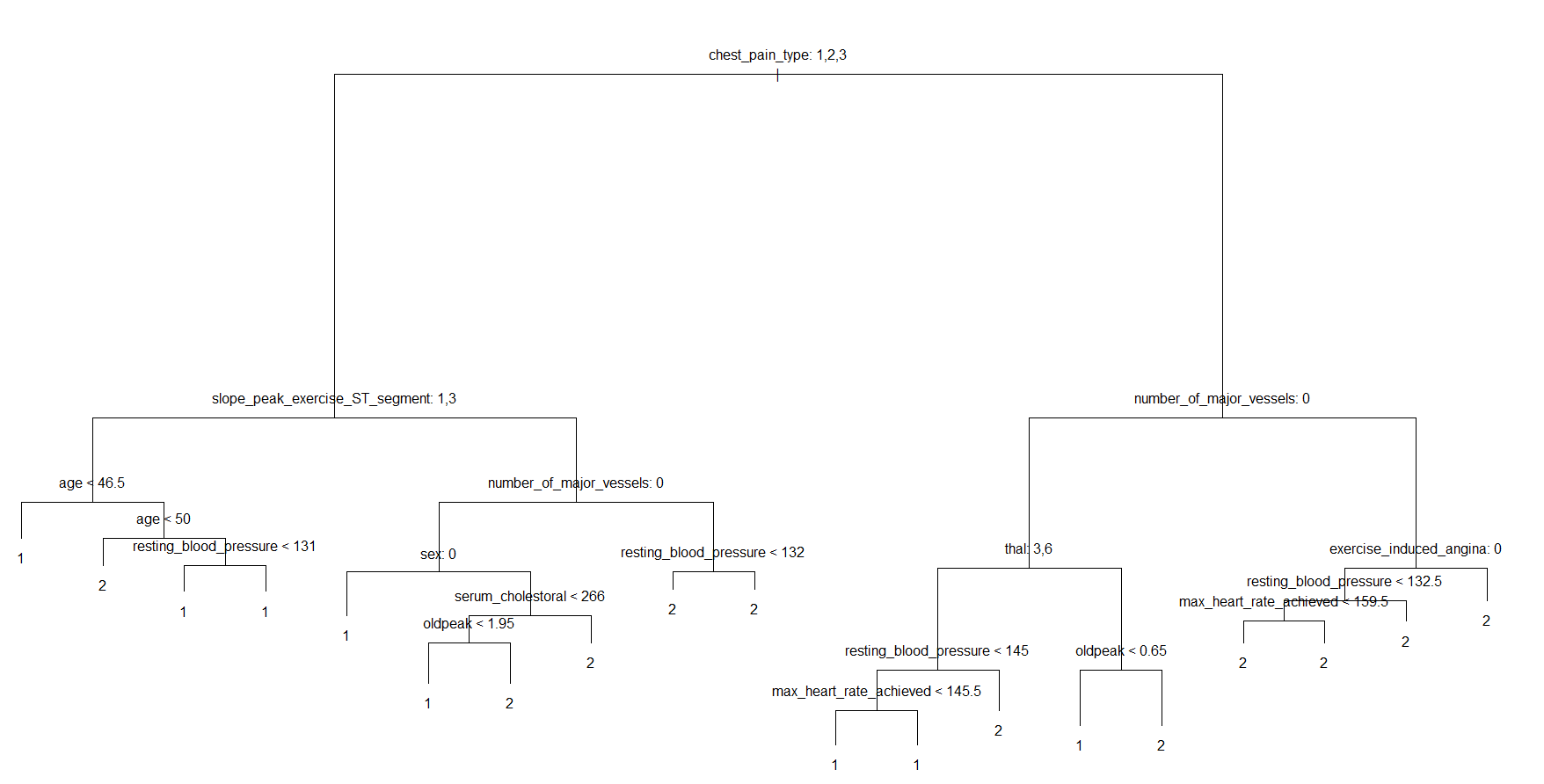
Overall accuracy is the sum of the diagonals in the confusion matrix divided by the number of rows in the test data set.



This is a good model, since overall accuracy (0.7) is greater than the baseline rate (0.54). This indicates that the splitting criteria used for creating the root & terminal nodes of a decision tree is correct. This approach could be used for extension of the terminal nodes, which in turn can create an accurate tree model.

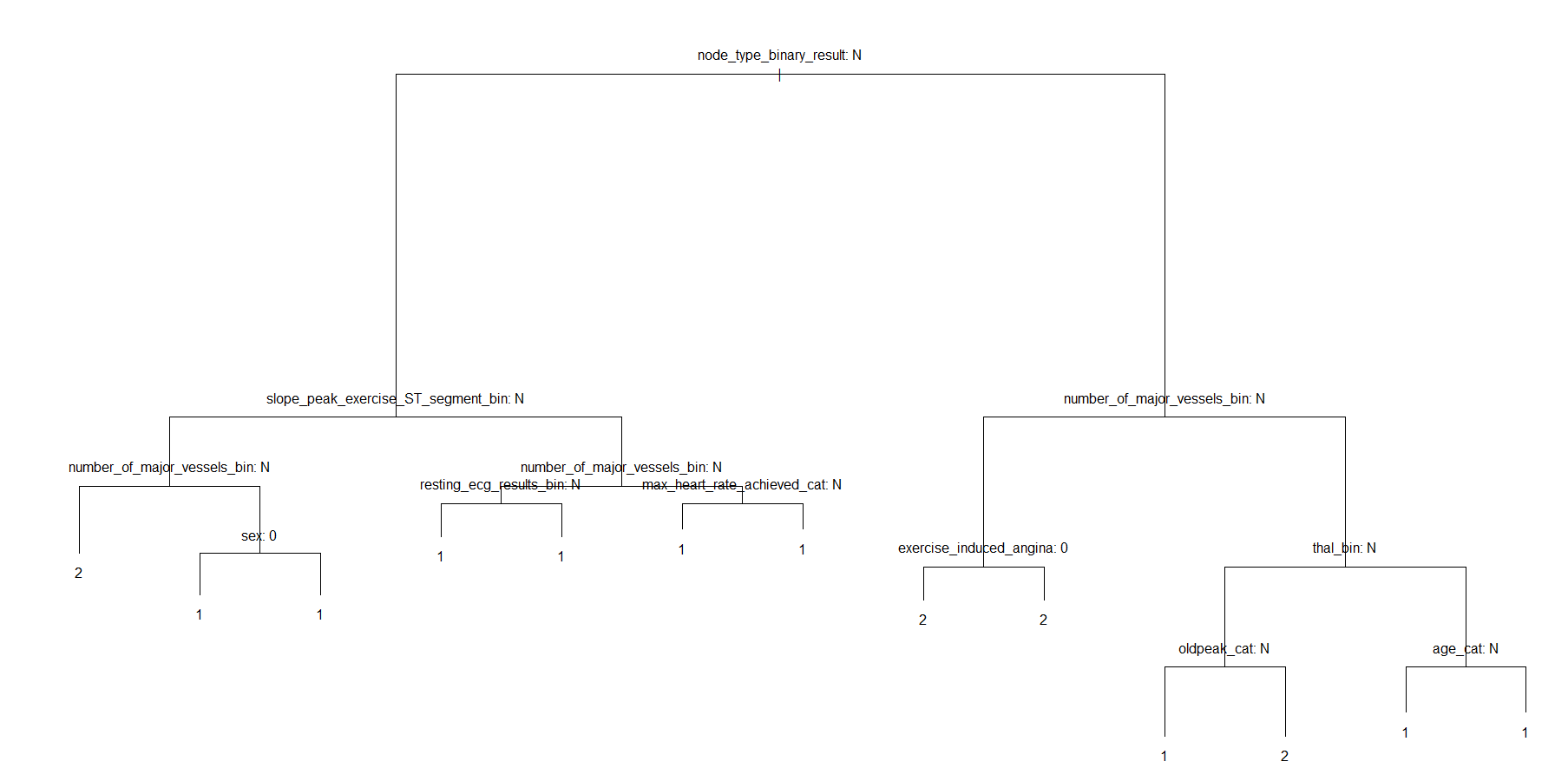
1. **Creating a tree using the tree package and comparing the results with the ones obtained in (8.).**

**Decision tree created for the heart data frame from the training data using the tree package:**

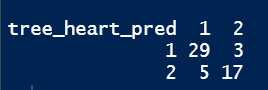


From the tree diagram, it can be said that the root node is "chest\_pain\_type" and the terminal nodes in the first split are "slope\_peak\_exercise\_ST\_segment" and "number\_of\_major\_vessels" for "chest\_pain\_type" values 1,2,3 and 4 respectively. This is identical to the results obtained in (5.) and (7.). The root node column "node\_type\_binary\_result" is the binary version of the original root node "chest\_pain\_type" and the terminal nodes "slope\_peak\_exercise\_ST\_segment\_bin" and "number\_of\_major\_vessels\_bin" are the binary versions of the original terminal nodes "slope\_peak\_exercise\_ST\_segment" and "number\_of\_major\_vessels". If "chest\_pain\_type" value is "4", then the "node\_type\_binary\_result" is "Yes", else the "node\_type\_binary\_result" is "No". This is because value "4" has the highest chance of occurrence. For the "No" value in "node\_type\_binary\_result" predictor, the terminal node would be the binary predictor "slope\_peak\_exercise\_ST\_segment\_bin". For the "Yes" value in "node\_type\_binary\_result" predictor, the terminal node would be the binary predictor "number\_of\_major\_vessels\_bin". This is verified by plotting the decision tree for the transformed training data.

**Decision tree created for the transformed training data using the tree package:**



**Confusion Matrix:**



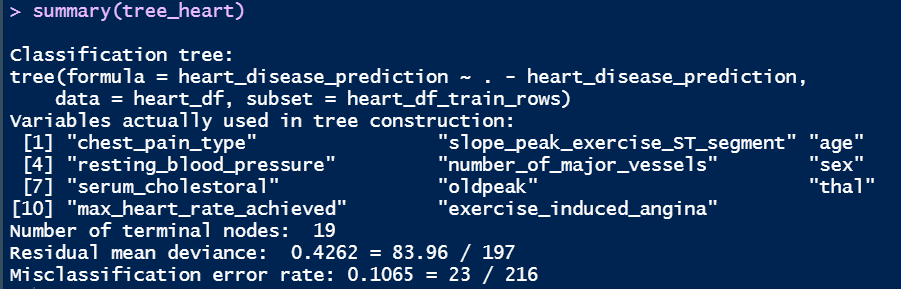
The rows represent the predicted values, while the columns represent the actual values.



The overall accuracy of the tree is 0.8518519.

Calculated overall accuracy (0.7) is lesser than the overall accuracy (0.85) from the tree. The difference is due to the algorithms used in the tree creation. The tree () function uses 'Gini' indexing as splitting criteria, while the calculated one uses Entropy and information gain.

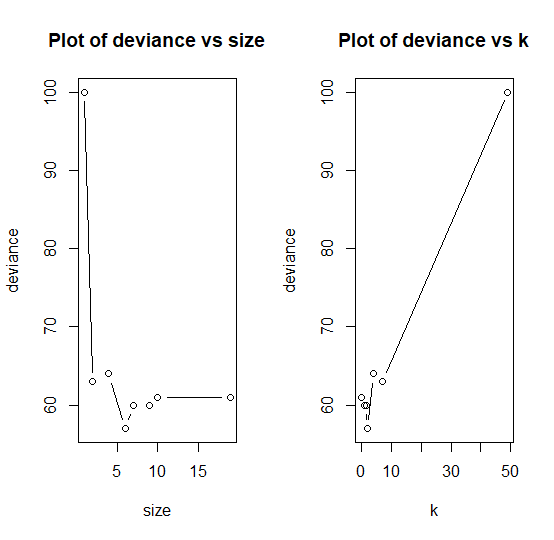
**Summary of the tree:**



The tree summary shows that there are 19 terminal nodes. This is a huge tree and there is a strong chance for misclassification. Hence, this tree could be pruned to check if there is any increase in model accuracy. It can be done using cross-fold validation techniques. Here, 19 is the best cross-validation error rate.

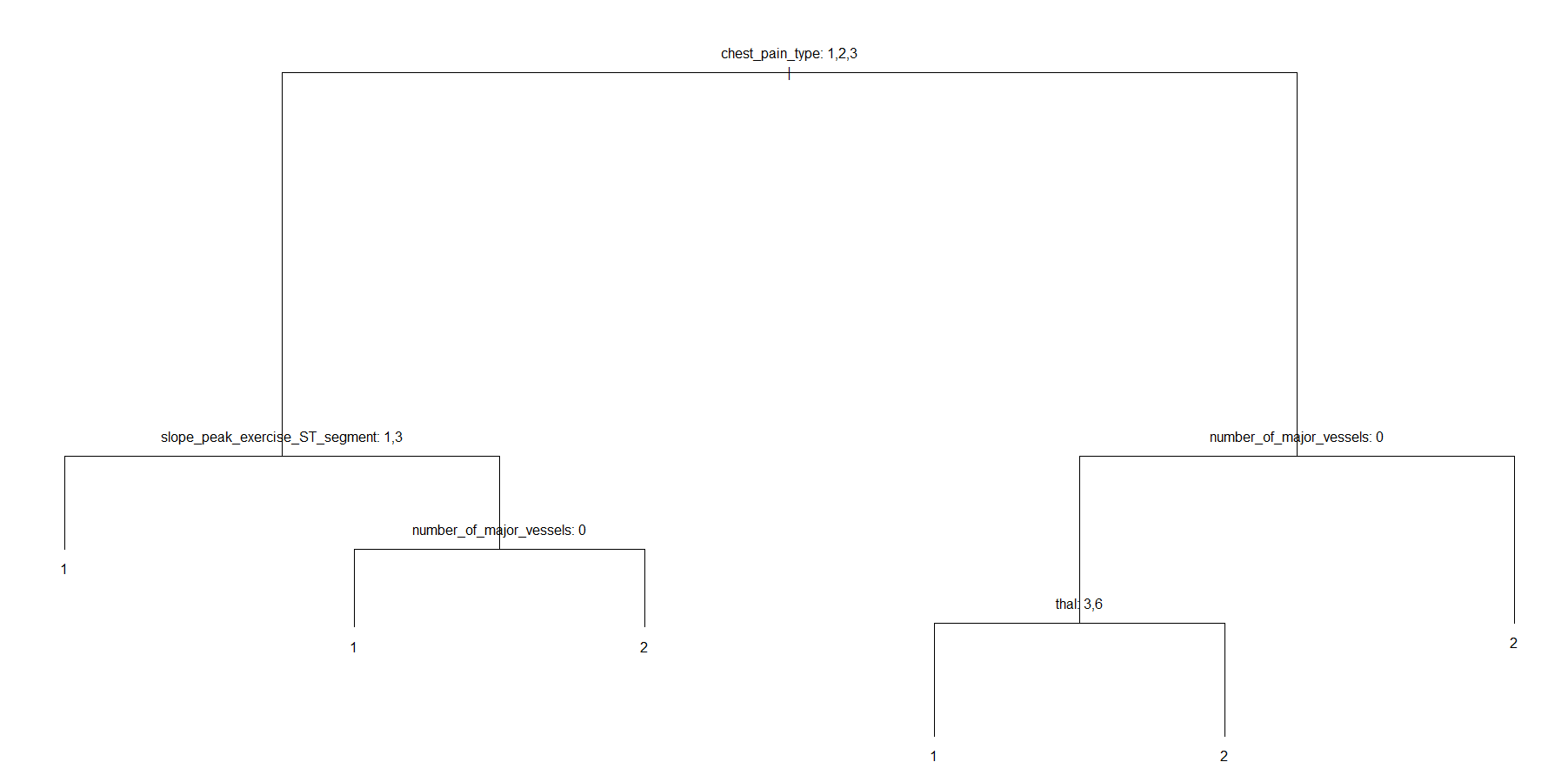
Here 5 folds of cross-validation is used.

**Plots of deviance vs size and deviance vs 'k' for the cross validated tree:**

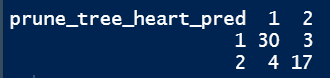


Size of tree = 6 looks best here as it holds the least deviance.

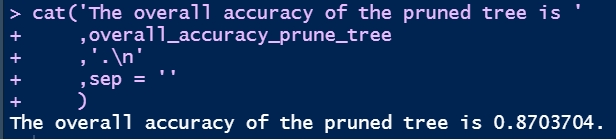
**Plot of the pruned tree with size = 6:**



**Confusion Matrix:**



The rows represent the predicted values, while the columns represent the actual values.



The overall accuracy of the pruned tree is 0.8703704.

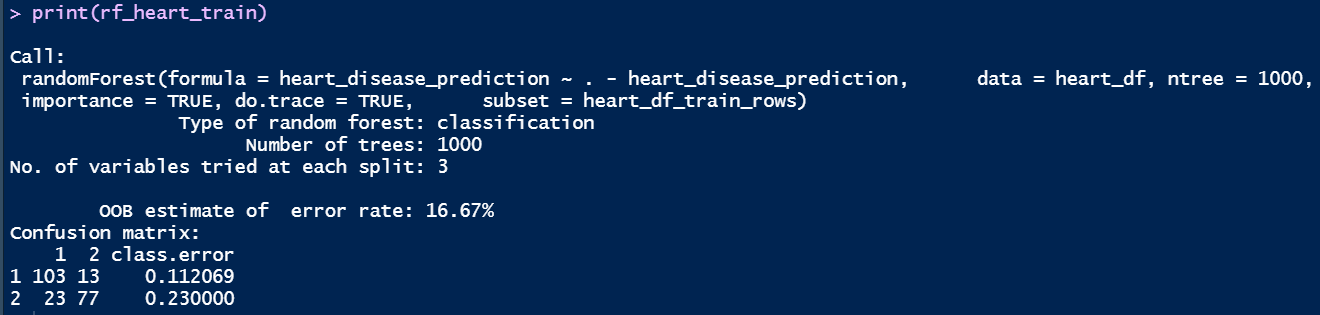
The test set error (overall accuracy) of the tree is 87%. Pruning the tree is a better option since the accuracy has increased from 85% to 87%. This would make a better model with more accurate predictions.

1. **Check for improvement in accuracy when random forest model is applied.**

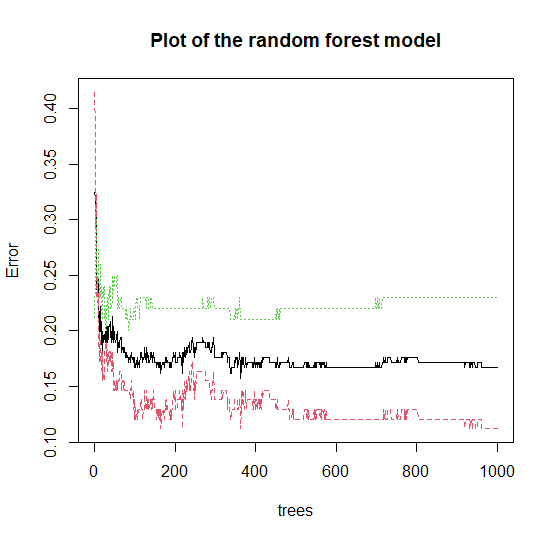
The main drawback of using decision trees is that they produce high variance. However, having models with low variance is necessary. Hence, bagging aka bootstrapping aggregation is to be used. Bootstrapping is re-sampling the observed data by randomly sampling (within itself) with replacement. The sample length will be same as that of the observation length. Due to this, plenty of training datasets would be created, which can then be averaged to reduce the variance. The trees are not pruned here, which means that each individual tree has high variance but low bias (because of random sampling). Averaging this reduces both the variance and the bias. The remaining data from the randomly selected data could be considered as the test data. This data is also known as Out-of-Bag data (OOB). The MSE of the OOB error is calculated and the model with lowest MSE is considered.

Here, the random forest algorithm is used on the training data set, with number of trees = 1000 and “mtry” value = root of the number of predictors.

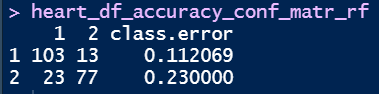
**Random forest simulation result:**



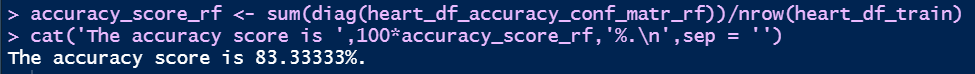
**Plot of the random forest model:**



**Confusion Matrix of Random forest model:**

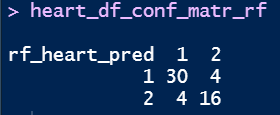


The rows represent the predicted values, while the columns represent the actual values.

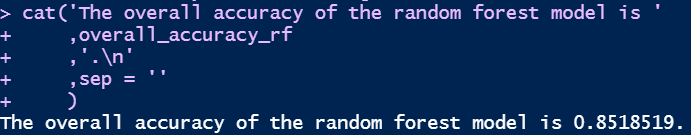


The accuracy score is 83.33333%.

**Confusion Matrix of predicted response from random forest model:**



The rows represent the predicted values, while the columns represent the actual values.



The overall accuracy of the random forest model is 0.8518519.

Random forest's overall accuracy (0.85) is lesser than the pruned tree's overall accuracy (0.87). The accuracy score (83.33) is also lesser than the pruned tree's overall accuracy (0.87). This means that the random forest technique need not be applied, as pruning is much more accurate. The accuracy of the pruned tree (0.87) is quite good. This itself could be used instead of going with random forest model.