



Heart Disease Prediction

AS a request for Data Mining & Predictive Analytics Course Requirements

Introduction

Motivation

Working on the heart disease dataset was motivated by the fact that heart disease claims millions of lives each year and is a major cause of death globally. Working with this dataset will help us understand the different risk factors that lead to heart disease and create models that can predict a person's chance of developing the condition in their lifetime.

Goal of the project

My goal on this project by using machine learning algorithms to predict the life hood of a person developing heart disease based on their demographic and health-related attributes. This could help doctors and healthcare providers identify high-risk patients early on and take preventive measures. By analyzing the heart disease dataset, we can gain insights into the various risk factors that contribute to heart disease, such as age, gender, painloc, cholesterol levels, and fasting blood sugar. This knowledge can be used to educate people about the importance of healthy lifestyle choices and early intervention. By understanding the factors that contribute to heart disease and predicting the life hood of developing heart disease, you can develop personalized treatment plans that are tailored to each patient's needs. This can improve treatment outcomes and reduce the risk of complications.

Why it is interesting

The heart disease dataset is multi-disciplinary, involving fields such as cardiology, epidemiology, and machine learning. This provides an opportunity to collaborate with experts from different fields and develop innovative solutions to address heart disease. Heart disease affects different populations differently, and by analyzing the heart disease dataset, we can identify healthcare disparities that exist and develop targeted interventions to address them. This can help reduce healthcare inequities and improve health outcomes for all.

Why machine learning is a reasonable approach

Machine learning is best approach of heart disease dataset contains many variables, and the relationships between these variables can be complex. Machine learning algorithms can identify patterns and relationships in the data that may not be apparent using traditional statistical methods. The relationships between variables in the heart disease dataset may not be linear, which makes traditional statistical methods less effective. Machine learning algorithms can model nonlinear relationships and identify complex interactions between variables. One of the main goals of analyzing the heart disease dataset is to predict the life hood of a person developing heart disease. Machine learning algorithms can be trained on the data to make accurate predictions, which can help healthcare providers identify high-risk patients and take preventive measures. Where machine learning is a reasonable approach for analyzing the heart disease dataset because it can create complex model relationships between variables and make accurate predictions for developing personalized treatment plans, and scale to handle large amounts of data.

Keywords associated with the project.

- Predicting risk attributes of heart diseases
- Data Analysis and pre-processing
- Machine learning algorithms
- Testing the models with test and trained data
- Predictive modeling

Background

Background information regarding the dataset and the overall message/information that will be conveyed.

The dataset contains 14 attributes and 303 records here the predict the heart diseases with machine learning algorithm we use patients.

- age
- sex
- cp (chest pain type)
- trestbps (resting blood pressure)
- chol (serum cholesterol in mg/dl)
- fbs (fasting blood sugar >120mg/dl(1=true;0=false)),
- restecg (resting electrocardiographic results [value 0 : normal, value 1: having ST-T wave abnormality (T wave inversions and/or ST elevation or depression of > 0.05 mV), Value 2: showing probable or definite left ventricular hypertrophy by Estes' criteria])
- Thalach (maximum heart rate achieved)
- exang(exercise induced angina (1 = yes; 0 = no))
- oldpeak (ST depression induced by exercise relative to rest)
- slope (the slope of the peak exercise ST segment [Value 1: upsloping, Value 2: flat, Value 3: down sloping])
- ca (number of major vessels (0-3) colored by fluoroscopy)
- thal: 3 = normal; 6 = fixed defect; 7 = reversible defect
- num: diagnosis of heart disease ((angiographic disease status [Value 0: $< 50\%$ diameter narrowing, Value 1: $> 50\%$ diameter narrowing (in any major vessel: attributes 59 through 68 are vessels)])).

Research question /problem to be addressed

Need to develop a model that can predict the presence or absence of heart disease based on these variables. So by using given variables we can accurately predict the presence or absence of heart disease in patients using demographic information, clinical measurements, and medical history data such as age, sex, chest pain type, resting blood pressure, cholesterol levels, fasting blood sugar levels, resting electrocardiographic results, maximum heart rate achieved, exercise induced angina, ST depression induced by exercise relative to rest, the slope of the peak exercise ST segment, number of major vessels colored by fluoroscopy, and thallium stress test results. These results need to be given by machine learning algorithms based on available data.

Is the available data sufficient to deliver the intended message?

Yes, we have sufficient data in the dataset to test and train the model.

Dataset Description:

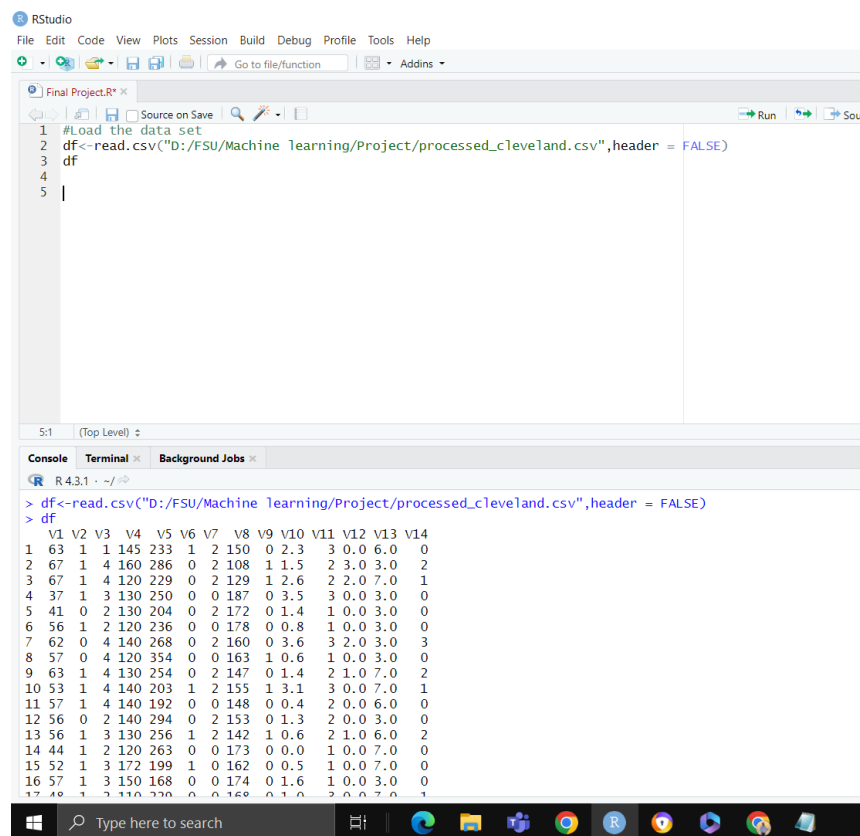
Description of the sources of data

The heart disease dataset is a publicly available dataset that was sourced from the UCI Machine Learning Repository. The dataset was created by researchers at the Cleveland Clinic Foundation in Ohio, USA, and contains data on patients who were referred to the clinic for chest pain evaluation. The data was collected between 1988 and 1990, and includes demographic information, clinical measurements, and medical history data for a total of 303 patients. The dataset was originally created to investigate the use of machine learning algorithms for predicting the presence or absence of heart disease based on patient data. It has since become a popular benchmark dataset for evaluating the performance of machine learning algorithms for classification tasks in healthcare.

Content of the dataset and its significance for project

The heart disease dataset contains important information about patient's chest pain evaluation. For me this dataset represents an opportunity to explore the use of machine learning algorithms. By analyzing the dataset and developing predictive models, I can gain insights into the risk factors for heart disease and identify patients who may be at high risk for developing the condition. This information can help healthcare providers to take pro-active measures to prevent heart disease and improve patient outcomes. Additionally, when I am working with this dataset this can help me to develop important skills in data analysis, statistical modeling, and machine learning.

Explore the data



```
#load the data set
df<-read.csv("D:/FSU/Machine Learning/Project/processed_cleveland.csv",header = FALSE)
df
```

	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	V13	V14
1	63	1	1	145	233	1	2	150	0	2.3	3	0.0	6.0	0
2	67	1	4	160	286	0	2	108	1	1.5	2	3.0	3.0	2
3	67	1	4	120	229	0	2	129	1	2.6	2	2.0	7.0	1
4	37	1	3	130	250	0	0	187	0	3.5	3	0.0	3.0	0
5	41	0	2	130	204	0	2	172	0	1.4	1	0.0	3.0	0
6	56	1	2	120	236	0	0	178	0	0.8	1	0.0	3.0	0
7	62	0	4	140	268	0	2	160	0	3.6	3	2.0	3.0	3
8	57	0	4	120	354	0	0	163	1	0.6	1	0.0	3.0	0
9	63	1	4	130	254	0	2	147	0	1.4	2	1.0	7.0	2
10	53	1	4	140	203	1	2	155	1	3.1	3	0.0	7.0	1
11	57	1	4	140	192	0	0	148	0	0.4	2	0.0	6.0	0
12	56	0	2	140	294	0	2	153	0	1.3	2	0.0	3.0	0
13	56	1	3	130	256	1	2	142	1	0.6	2	1.0	6.0	2
14	44	1	2	120	263	0	0	173	0	0.0	1	0.0	7.0	0
15	52	1	3	172	199	1	0	162	0	0.5	1	0.0	7.0	0
16	57	1	3	150	168	0	0	174	0	1.6	1	0.0	3.0	0
17	48	1	2	110	220	0	0	168	0	1.0	2	0.0	7.0	1

Does the data set need some cleaning or customization in order to fit with your model's requirements?

Yes, I have cleaned the dataset. I have renamed the column names with proper names which are given in dataset description. So now, I can easily test-train and dataset and can create exact model using machine learning algorithms.

Model's Baseline

Description of the machine learning algorithm that we will use to build the model

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model. Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset. Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output. The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.

Assumptions for Random Forest:

Since the random forest combines multiple trees to predict the class of the dataset, it is possible that some decision trees may predict the correct output, while others may not. But together, all the trees predict the correct output. Therefore, below are two assumptions for a better Random Forest classifier.

- There should be some actual values in the feature variable of the dataset so that the classifier can predict accurate results rather than a guessed result.
- The predictions from each tree must have very low correlations.

Why preferred this algorithm?

- It takes less training time as compared to other algorithms.
- It predicts output with high accuracy, even for the large dataset it runs efficiently.
- It can also maintain accuracy when a large proportion of data is missing.

Pros and cons associated with the chosen algorithm

Pros:

- Random Forest can perform both Classification and Regression tasks.
- It is capable of handling large datasets with high dimensionality.
- It enhances the accuracy of the model and prevents the overfitting issue.

Cons:

- Random forests can still overfit the data, especially when the number of trees in the forest is too large or when the model is too complex.
- Random forests can be difficult to interpret, as the model outputs are not easily explainable.
- Random forests can perform poorly on datasets with imbalanced classes, where one class is significantly underrepresented in the data.

Building the machine learning model:

Step 1: Import required libraries.

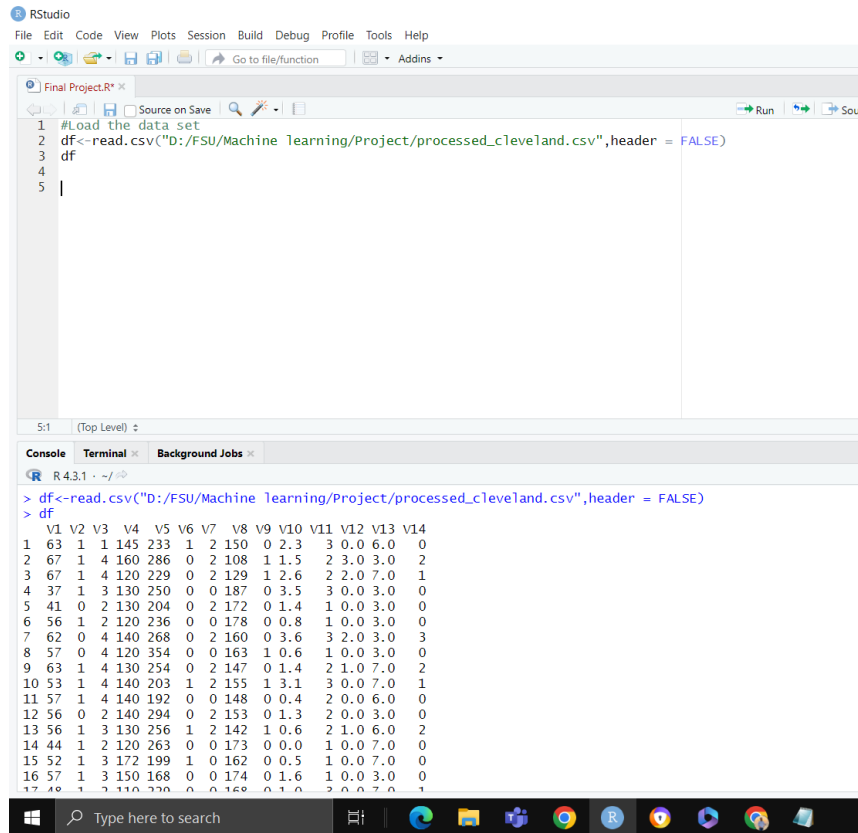
```
library("caret")
```

```
library("class")
```

```
install.packages("randomForest")
```

```
library(randomForest)
```

Step 2: Load the dataset.

A screenshot of the RStudio interface. The top menu bar includes File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, and Help. Below the menu is a toolbar with icons for file operations and a search bar. The main editor window shows a script with the following code:

```
1 #Load the data set
2 df<-read.csv("D:/FSU/Machine learning/Project/processed_cleveland.csv",header = FALSE)
3 df
4
5 |
```

The console window at the bottom shows the execution of the code:

```
> df<-read.csv("D:/FSU/Machine learning/Project/processed_cleveland.csv",header = FALSE)
> df
  V1 V2 V3 V4 V5 V6 V7 V8 V9 V10 V11 V12 V13 V14
1 63 1 1 145 233 1 2 150 0 2.3 3 0.0 6.0 0
2 67 1 4 160 286 0 2 108 1 1.5 2 3.0 3.0 2
3 67 1 4 120 229 0 2 129 1 2.6 2 2.0 7.0 1
4 37 1 3 130 250 0 0 187 0 3.5 3 0.0 3.0 0
5 41 0 2 130 204 0 2 172 0 1.4 1 0.0 3.0 0
6 56 1 2 120 236 0 0 178 0 0.8 1 0.0 3.0 0
7 62 0 4 140 268 0 2 160 0 3.6 3 2.0 3.0 3
8 57 0 4 120 354 0 0 163 1 0.6 1 0.0 3.0 0
9 63 1 4 130 254 0 2 147 0 1.4 2 1.0 7.0 2
10 53 1 4 140 203 1 2 155 1 3.1 3 0.0 7.0 1
11 57 1 4 140 192 0 0 148 0 0.4 2 0.0 6.0 0
12 56 0 2 140 294 0 2 153 0 1.3 2 0.0 3.0 0
13 56 1 3 130 256 1 2 142 1 0.6 2 1.0 6.0 2
14 44 1 2 120 263 0 0 173 0 0.0 1 0.0 7.0 0
15 52 1 3 172 199 1 0 162 0 0.5 1 0.0 7.0 0
16 57 1 3 150 168 0 0 174 0 1.6 2 0.0 3.0 0
17 48 1 2 110 220 0 0 168 0 1.0 2 0.0 7.0 1
```

The bottom status bar shows the R version 4.3.1 and the current file path.

First, I have loaded the dataset. Once the dataset is loaded, I have checked for missing values and there are no missing values in the dataset. Later using dataset description, I have renamed the columns with actual column names.

```
RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function Addins
Final Project.R
Source on Save Run Source
5 #Renamed Columns
6 colnames(df)<-c("age","sex","cp","trestbps","chol","fbs","restecg",
7               "thalach","exang","oldpeak","slope","ca","thal","num")
8 colnames(df)
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
10:1 (Top Level) R Sc

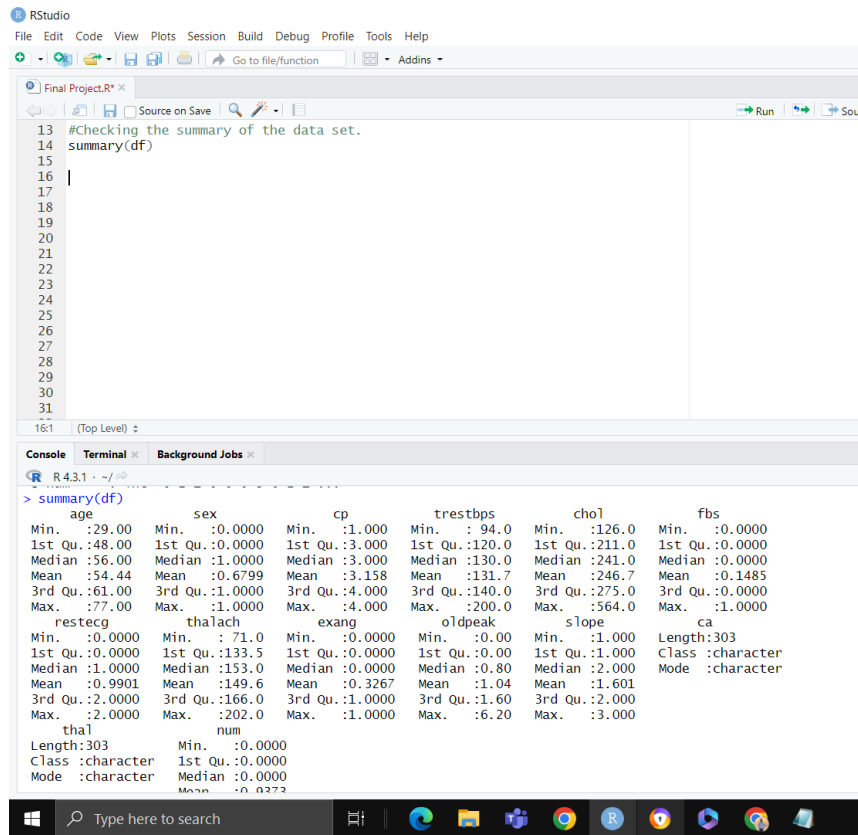
Console Terminal Background Jobs
R 4.3.1 ~ /
60 51 1 1 125 213 0 2 125 1 1.4 1 1.0 3.0 0
61 51 0 4 130 305 0 0 142 1 1.2 2 0.0 7.0 2
62 46 0 3 142 177 0 2 160 1 1.4 3 0.0 3.0 0
63 58 1 4 128 216 0 2 131 1 2.2 2 3.0 7.0 1
64 54 0 3 135 304 1 0 170 0 0.0 1 0.0 3.0 0
65 54 1 4 120 188 0 0 113 0 1.4 2 1.0 7.0 2
66 60 1 4 145 282 0 2 142 1 2.8 2 2.0 7.0 2
67 60 1 3 140 185 0 2 155 0 3.0 2 0.0 3.0 1
68 54 1 3 150 232 0 2 165 0 1.6 1 0.0 7.0 0
69 59 1 4 170 326 0 2 140 1 3.4 3 0.0 7.0 2
70 46 1 3 150 231 0 0 147 0 3.6 2 0.0 3.0 1
71 65 0 3 155 269 0 0 148 0 0.8 1 0.0 3.0 0
[ reached 'max' / getOption("max.print") -- omitted 232 rows ]
> colnames(df)<-c("age","sex","cp","trestbps","chol","fbs","restecg",
+               "thalach","exang","oldpeak","slope","ca","thal","num")
> colnames(df)
[1] "age" "sex" "cp" "trestbps" "chol" "fbs" "restecg" "thalach" "exang"
[10] "oldpeak" "slope" "ca" "thal" "num"
>
```

Step 3: Check the structure of the dataset.

```
RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function Addins
Final Project.R
Source on Save Run Source
10 #Check the structure of the dataset.
11 str(df)
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
13:1 (Top Level)

Console Terminal Background Jobs
R 4.3.1 ~ /
[1] "age" "sex" "cp" "trestbps" "chol" "fbs" "restecg" "thalach" "exang"
[10] "oldpeak" "slope" "ca" "thal" "num"
> str(df)
'data.frame': 303 obs. of 14 variables:
 $ age : num 63 67 67 37 41 56 62 57 63 53 ...
 $ sex : num 1 1 1 1 0 1 0 0 1 1 ...
 $ cp : num 1 4 4 3 2 2 4 4 4 4 ...
 $ trestbps: num 145 160 120 130 130 120 140 120 130 140 ...
 $ chol : num 233 286 229 250 204 236 268 354 254 203 ...
 $ fbs : num 1 0 0 0 0 0 0 0 1 ...
 $ restecg: num 2 2 2 0 2 0 2 0 2 2 ...
 $ thalach: num 150 108 129 187 172 178 160 163 147 155 ...
 $ exang : num 0 1 1 0 0 0 1 0 1 ...
 $ oldpeak: num 2.3 1.5 2.6 3.5 1.4 0.8 3.6 0.6 1.4 3.1 ...
 $ slope : num 3 2 2 3 1 1 3 1 2 3 ...
 $ ca : chr "0.0" "3.0" "2.0" "0.0" ...
 $ thal : chr "6.0" "3.0" "7.0" "3.0" ...
 $ num : int 0 2 1 0 0 0 3 0 2 1 ...
>
```

Step 4: Checking the summary.



The screenshot shows the RStudio interface. The script editor contains the following code:

```
13 #Checking the summary of the data set.
14 summary(df)
15
16 |
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
```

The console output shows the summary of the dataset 'df':

```
> summary(df)
 age          sex          cp          trestbps          chol          fbs
Min.   :29.00  Min.   :0.0000  Min.   :1.000  Min.   : 94.0  Min.   :126.0  Min.   :0.0000
1st Qu.:48.00  1st Qu.:0.0000  1st Qu.:3.000  1st Qu.:120.0  1st Qu.:211.0  1st Qu.:0.0000
Median :56.00  Median :1.0000  Median :3.000  Median :130.0  Median :241.0  Median :0.0000
Mean   :54.44  Mean   :0.6799  Mean   :3.158  Mean   :131.7  Mean   :246.7  Mean   :0.1485
3rd Qu.:61.00  3rd Qu.:1.0000  3rd Qu.:4.000  3rd Qu.:140.0  3rd Qu.:275.0  3rd Qu.:0.0000
Max.   :77.00  Max.   :1.0000  Max.   :4.000  Max.   :200.0  Max.   :564.0  Max.   :1.0000

 restecg      thalach      exang      oldpeak      slope      ca
Min.   :0.0000  Min.   : 71.0  Min.   :0.0000  Min.   :0.00  Min.   :1.000  Length:303
1st Qu.:0.0000  1st Qu.:133.5  1st Qu.:0.0000  1st Qu.:0.00  1st Qu.:1.000  Class :character
Median :1.0000  Median :133.0  Median :0.0000  Median :0.80  Median :2.000  Mode :character
Mean   :0.9901  Mean   :149.6  Mean   :0.3267  Mean   :1.04  Mean   :1.601
3rd Qu.:2.0000  3rd Qu.:166.0  3rd Qu.:1.0000  3rd Qu.:1.60  3rd Qu.:2.000
Max.   :2.0000  Max.   :202.0  Max.   :1.0000  Max.   :6.20  Max.   :3.000

 thal      num
Length:303  Min.   :0.0000
Class :character  1st Qu.:0.0000
Mode :character  Median :0.0000
                  Mean   :0.0373
```

As I am working on heart diseases data. So, the dataset contains as column called “num” it nothing but predictive attribute of patient heart disease. So, this column has only 4 values that is 0,1,2,3,4 it is range of heart diseases as per dataset description folder here ‘0’ is patient has no heart diseases,’1,2,3’ ranges of normal pain or normal heart diseases and 4 ranges for critical heart pain or disease. So, when tried to test and train the data by creating as data portion or splitting it has failed because it was able the “num” column predictive attribute has range of 0 to 4 so machine was unable to the split the data. So, I have used function called “ifelse” Where it can convert or make values in an easy way that machine can understand e.g.: we have as per description. “0” means patients without heart problems and 1,2,3,4 has various heart problems conditions. So here “ifelse” function has converted that 2,3,4 ranges is equal to 1 and 0 equal to 0. Now, I have renamed “num” to predicted attribute and we drop the column “num” as well.

```
> df$predicted_attribute<-ifelse(df$num>0,1,0)
> df$predicted_attribute
 [1] 0 1 1 0 0 0 1 0 1 1 0 0 1 0 0 0 1 0 0 0 0 0 1 1 1 0 0 0 0 1 0 1 1 0 0 0 0 1 1 1 0 1 0 0 0 0 1 1 0 1 0 0 0 0 1 1 0 1 0 0 0 0
[53] 1 0 1 1 1 1 0 0 1 0 1 0 1 1 1 0 1 1 1 0 1 1 0 1 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 1 1 1 0 0 0 0 0 0
[105] 1 0 1 1 1 1 1 0 1 1 0 0 1 1 1 0 1 1 0 1 1 0 0 0 0 0 0 0 1 1 1 0 0 1 0 1 0 1 1 0 0 0 0 0 0 1 1 1
[157] 1 1 1 0 0 1 0 0 0 0 0 0 1 0 1 0 1 1 0 1 0 0 1 1 0 0 1 0 0 0 1 1 1 0 1 1 1 0 1 0 0 0 1 0 0 0 0 0 1 1 1
[209] 0 1 0 1 0 1 1 0 0 0 0 0 0 0 1 1 0 0 0 1 1 0 0 1 1 0 1 1 0 0 0 0 0 1 0 1 1 1 1 0 0 1 0 0 0 0 0 0 0 1
[261] 0 1 0 0 1 1 1 1 0 1 0 1 0 1 0 0 0 1 0 1 0 1 0 1 1 1 0 0 0 1 0 1 1 1 0 1 1 1 1 1 1 0
>
```

```
> df<-subset(df,select = -c(num))
> df
  age sex cp trestbps chol fbs restecg thalach exang oldpeak slope ca thal predicted_attribute
1  63  1  1   145  233   1     2   150    0    2.3   3 0.0  6.0             0
2  67  1  4   160  286   0     2   108    1    1.5   2 3.0  3.0             1
3  67  1  4   120  229   0     2   129    1    2.6   2 2.0  7.0             1
4  37  1  3   130  250   0     0   187    0    3.5   3 0.0  3.0             0
5  41  0  2   130  204   0     2   172    0    1.4   1 0.0  3.0             0
6  56  1  2   120  236   0     0   178    0    0.8   1 0.0  3.0             0
7  62  0  4   140  268   0     2   160    0    3.6   3 2.0  3.0             1
8  57  0  4   120  354   0     0   163    1    0.6   1 0.0  3.0             0
9  63  1  4   130  254   0     2   147    0    1.4   2 1.0  7.0             1
10 53  1  4   140  203   1     2   155    1    3.1   3 0.0  7.0             1
11 57  1  4   140  192   0     0   148    0    0.4   2 0.0  6.0             0
12 56  0  2   140  294   0     2   153    0    1.3   2 0.0  3.0             0
13 56  1  3   130  256   1     2   142    1    0.6   2 1.0  6.0             1
14 44  1  2   120  263   0     0   173    0    0.0   1 0.0  7.0             0
15 52  1  3   172  199   1     0   162    0    0.5   1 0.0  7.0             0
16 57  1  3   150  168   0     0   174    0    1.6   1 0.0  3.0             0
17 48  1  2   110  229   0     0   168    0    1.0   3 0.0  7.0             1
```


Step 5: Train - Test Split.

```
RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Final Project.R* x
Source on Save Run Source
24
25 #Train - test split
26 set.seed(123)
27 trainindex<-createDataPartition(df$predicted_attribute,p=0.7,list = FALSE,times = 1)
28 trainindex
29
30
31
32
33
34
35
36
37
38
39
40
41
42
30:1 (Top Level) R Sc
```

```
Console Terminal Background Jobs x
R 4.3.1 - ~/
> set.seed(123)
> trainindex<-createDataPartition(df$predicted_attribute,p=0.7,list = FALSE,times = 1)
> trainindex
Resample1
[1,] 1
[2,] 4
[3,] 5
[4,] 6
[5,] 7
[6,] 9
[7,] 10
[8,] 11
[9,] 13
[10,] 14
[11,] 16
[12,] 17
[13,] 20
[14,] 21
[15,] 22
[16,] 23
```

```
RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Final Project.R* x
Source on Save Run Source
29
30 trainData<-df[trainindex,]
31 trainData
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
33:1 (Top Level) R Sc
```

```
Console Terminal Background Jobs x
R 4.3.1 - ~/
> trainData<-df[trainindex,]
> trainData
  age sex cp trestbps chol fbs restecg thalach exang oldpeak slope ca thal predicted_attribute
1  63  1  1   145  233  1     2    150  0     2.3   3 0.0  6.0             0
4  37  1  3   130  250  0     0    187  0     3.5   3 0.0  3.0             0
5  41  0  2   130  204  0     2    172  0     1.4   1 0.0  3.0             0
6  56  1  2   120  236  0     0    178  0     0.8   1 0.0  3.0             0
7  62  0  4   140  268  0     2    160  0     3.6   3 2.0  3.0             1
9  63  1  4   130  254  0     2    147  0     1.4   2 1.0  7.0             1
10 53  1  4   140  203  1     2    155  1     3.1   3 0.0  7.0             1
11 57  1  4   140  192  0     0    148  0     0.4   2 0.0  6.0             0
13 56  1  3   130  256  1     2    142  1     0.6   2 1.0  6.0             1
14 44  1  2   120  263  0     0    173  0     0.0   1 0.0  7.0             0
16 57  1  3   150  168  0     0    174  0     1.6   1 0.0  3.0             0
17 48  1  2   110  229  0     0    168  0     1.0   3 0.0  7.0             1
20 49  1  2   130  266  0     0    171  0     0.6   1 0.0  3.0             0
21 64  1  1   110  211  0     2    144  1     1.8   2 0.0  3.0             0
22 58  0  1   150  283  1     2    162  0     1.0   1 0.0  3.0             0
23 58  1  2   120  284  0     2    160  0     1.8   2 0.0  3.0             1
24 58  1  3   132  224  0     2    173  0     3.2   1 2.0  7.0             1
```

The screenshot shows the RStudio interface. The script editor contains the following code:

```

32
33 testData<-df[-trainindex,]
34 testData
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50

```

The console shows the execution of the code, displaying the structure and first few rows of the `testData` dataset:

```

> testData<-df[-trainindex,]
> testData
  age sex cp trestbps chol fbs restecg thalach exang oldpeak slope ca thal predicted_attribute
2  67  1  4   160  286  0      2    108    1    1.5    2  3.0  3.0                1
3  67  1  4   120  229  0      2    129    1    2.6    2  2.0  7.0                1
8  57  0  4   120  354  0      0    163    1    0.6    1  0.0  3.0                0
12 56  0  2   140  294  0      2    153    0    1.3    2  0.0  3.0                0
15 52  1  3   172  199  1      0    162    0    0.5    1  0.0  7.0                0
18 54  1  4   140  239  0      0    160    0    1.2    1  0.0  3.0                0
19 48  0  3   130  275  0      0    139    0    0.2    1  0.0  3.0                0
28 66  0  1   150  226  0      0    114    0    2.6    3  0.0  3.0                0
31 69  0  1   140  239  0      0    151    0    1.8    1  2.0  3.0                0
37 43  1  4   120  177  0      2    120    1    2.5    2  0.0  7.0                1
38 57  1  4   150  276  0      2    112    1    0.6    2  1.0  6.0                1
44 59  1  3   150  212  1      0    157    0    1.6    1  0.0  3.0                0
45 61  0  4   130  330  0      2    169    0    0.0    1  0.0  3.0                1
47 51  1  3   110  175  0      0    123    0    0.6    1  0.0  3.0                0
49 65  0  3   140  417  1      2    157    0    0.8    1  1.0  3.0                0
56 54  1  4   124  266  0      2    109    1    2.2    2  1.0  7.0                1
62 46  0  3   142  177  0      2    160    1    1.4    3  0.0  3.0                0

```

First, we set the seed for reproducibility using the `set.seed()` function. Next, we used the `createDataPartition()` function to split the predictive attribute variable (which represents the target variable) into a training set with 70% of the data and a testing set with 30% of the data. The `list` argument is set to `FALSE` to return a logical vector, and the `times` argument is set to 1 to perform the splitting operation only once. Finally, we create the train Data and test Data datasets using the resulting indices.

Step 6: Separate the test labels from the test data.

First, we need to extract the target variable (predicted attribute) from the test Data using the `which()` and `names()` functions to find the column index of the predictive attribute column. Then we will use the negative indexing (`-which()`) to remove the predicted attribute column from the test Data. The extracted target variable is stored in the test Labels variable. After executing the code, test Data will contain all the columns except the predictive attribute column, and test Labels will contain the values of the predicted attribute column for each row in the test data.

```

RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Final Project.R*
# Separate the test labels from the test data.
test_labels<-testData$predicted_attribute
test_labels
test_Data<-testData[, -which(names(testData)== "predicted_attribute")]
test_Data

```

```

R 4.3.1 ~ /
> test_Data<-testData[, -which(names(testData)== "predicted_attribute")]
> test_Data
  age sex cp trestbps chol fbs restecg thalach exang oldpeak slope ca thal
2   67  1  4    160  286   0      2    108     1     1.5    2  3.0  3.0
3   67  1  4    120  229   0      2    129     1     2.6    2  2.0  7.0
8   57  0  4    120  354   0      0    163     1     0.6    1  0.0  3.0
12  56  0  2    140  294   0      2    153     0     1.3    2  0.0  3.0
15  52  1  3    172  199   1      0    162     0     0.5    1  0.0  7.0
18  54  1  4    140  239   0      0    160     0     1.2    1  0.0  3.0
19  48  0  3    130  275   0      0    139     0     0.2    1  0.0  3.0
28  66  0  1    150  226   0      0    114     0     2.6    3  0.0  3.0
31  69  0  1    140  239   0      0    151     0     1.8    1  2.0  3.0
37  43  1  4    120  177   0      2    120     1     2.5    2  0.0  7.0
38  57  1  4    150  276   0      2    112     1     0.6    2  1.0  6.0
44  59  1  3    150  212   1      0    157     0     1.6    1  0.0  3.0
45  61  0  4    130  330   0      2    169     0     0.0    1  0.0  3.0
47  51  1  3    110  175   0      0    123     0     0.6    1  0.0  3.0
49  65  0  3    140  417   1      2    157     0     0.8    1  1.0  3.0
56  54  1  4    124  266   0      2    109     1     2.2    2  1.0  7.0
62  46  0  3    142  177   0      2    160     1     1.4    3  0.0  3.0

```

Step 7: Train the model.

We use the random Forest () function to train a Random Forest model. The predicted attribute ~ . formula specifies that the num variable should be predicted using all other variables in the dataset. The data argument is set to the train Data variable containing the training data.

```

RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Final Project.R*
#Train the model.
install.packages("randomForest")
library(randomForest)
rfModel<-randomForest(predicted_attribute~.,data=trainData)
rfModel

```

```

R 4.3.1 ~ /
margin

Warning message:
package 'randomForest' was built under R version 4.3.2
> rfModel<-randomForest(predicted_attribute~.,data=trainData)
Warning message:
In randomForest.default(m, y, ...) :
The response has five or fewer unique values. Are you sure you want to do regression?
> rfModel

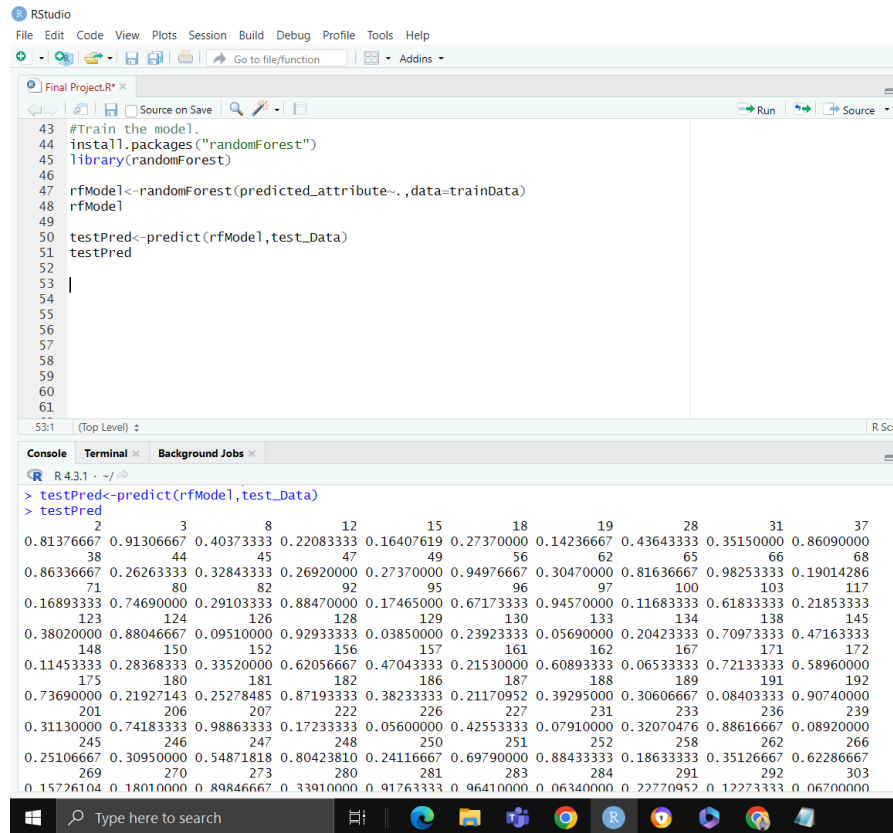
Call:
randomForest(formula = predicted_attribute ~ ., data = trainData)
Type of random forest: regression
Number of trees: 500
No. of variables tried at each split: 4

Mean of squared residuals: 0.1452551
% Var explained: 41.61
> |

```

The model was trained using 500 trees and tried 4 variables at each split. The "Mean of squared residuals" is a measure of the average distance between the predicted values and the actual values in the training data. A lower value indicates a better fit to the data. The "% Var explained" is a measure of how much of the variance in the outcome variable is explained by the predictor variables in the model. A higher value indicates a better fit to the data. In this case, the model has a mean squared residual of 0.1452551, which is relatively low, indicating a good fit to the training data. The % Var explained is 41.61%, which means that the predictor variables in the model explain 41.61% of the variance in the predicted attribute.

Step 8: Make predictions.



The screenshot shows the RStudio interface. The script editor contains the following code:

```
43 #Train the model.  
44 install.packages("randomForest")  
45 library(randomForest)  
46  
47 rfModel<-randomForest(predicted_attribute~.,data=trainData)  
48 rfModel  
49  
50 testPred<-predict(rfModel,test_Data)  
51 testPred  
52  
53 |  
54  
55  
56  
57  
58  
59  
60  
61
```

The console shows the output of the `testPred` command, displaying a matrix of predicted probabilities for 37 test instances. The first column represents the probability of not having heart disease (labeled as "0"), and the second column represents the probability of having heart disease (labeled as "1").

Instance	0 (Not having heart disease)	1 (Having heart disease)
1	0.81376667	0.91306667
2	0.40373333	0.22083333
3	0.16407619	0.27370000
4	0.14236667	0.43643333
5	0.35150000	0.86090000
6	0.86336667	0.26263333
7	0.32843333	0.26920000
8	0.27370000	0.94976667
9	0.30470000	0.81636667
10	0.98253333	0.19014286
11	0.16893333	0.74690000
12	0.29103333	0.88470000
13	0.17465000	0.67173333
14	0.94570000	0.11683333
15	0.61833333	0.21853333
16	0.38020000	0.88046667
17	0.09510000	0.92933333
18	0.03850000	0.23923333
19	0.05690000	0.20423333
20	0.70973333	0.47163333
21	0.11453333	0.28368333
22	0.33520000	0.62056667
23	0.47043333	0.21530000
24	0.60893333	0.06533333
25	0.72133333	0.58960000
26	0.73690000	0.21927143
27	0.25278485	0.87193333
28	0.38233333	0.21170952
29	0.39295000	0.30606667
30	0.08403333	0.90740000
31	0.31130000	0.74183333
32	0.98863333	0.17233333
33	0.05600000	0.42553333
34	0.07910000	0.32070476
35	0.88616667	0.08920000
36	0.25106667	0.30950000
37	0.54871818	0.80423810
38	0.24116667	0.69790000
39	0.88433333	0.18633333
40	0.35126667	0.62286667
41	0.15726104	0.18010000
42	0.89846667	0.33910000
43	0.91763333	0.96410000
44	0.06340000	0.22770952
45	0.12273333	0.06700000

The `testpred()` function to make predictions on the test data using the trained Random Forest model. The resulting predictions are stored in the `testPred` variable.

Step 9: Compare the predicted and actual values.

The predicted probabilities for each test instance of having heart disease (labeled as "1") or not (labeled as "0"). The first column represents the predicted probability for the instance not having heart disease, and the second column represents the predicted probability for the instance having heart disease: The first row shows a predicted probability of 0.0385 for not having heart disease and 0.9615 for having heart disease. The model predicts that this instance is more likely to have heart disease. The predictions can be evaluated against the true labels of the test set to assess the accuracy of the model. If the predicted probabilities are close to the true labels, the model is performing well. However, without knowing the true labels, it is difficult to interpret these predicted probabilities and what they mean for the heart disease dataset.

RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

Final Project.R* x

Source on Save Run

```
52
53 #Compare the predicted and actual values.
54 confMatrix<-table(testPred,test_labels)
55 confMatrix
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
```

57:1 (Top Level) ↕

Console Terminal Background Jobs

R 4.3.1 ~ /

```
> confMatrix<-table(testPred,test_labels)
> confMatrix
```

	test_labels
testPred	0 1
0.0385	1 0
0.056	1 0
0.0569	1 0
0.0634	1 0
0.06533333333333333	1 0
0.067	1 0
0.0791	1 0
0.08403333333333333	1 0
0.0892	1 0
0.0951	1 0
0.11453333333333333	1 0
0.11683333333333333	1 0
0.12273333333333333	1 0
0.14236666666666667	1 0
0.157261038961039	0 1
0.164076190476191	1 0

Type here to search

Conclusion:

The random forest algorithm used in this project produced a model that explained approximately 41.6% of the variance in the predicted attribute. While the predicted probabilities generated by the model can be evaluated against the true labels to assess accuracy, further analysis would be necessary to determine the overall performance of the model. Overall, the heart disease dataset and machine learning algorithms offer a promising approach to predicting and preventing heart disease, which is an important public health concern worldwide.

Struggles and difficulties using RStudio while building the model

Here, the difficulties are using R-studio is merging the “num” column values into 0 and 1 to find out the use “ifelse” function I have struggled to test and train to split the dataset.

Can you suggest other software(s) that work properly with your model? Why?

Yes, Python may be the best software for developing the models because it has extensive library options and easy way of coding it has pandas and pyspark coding choices.

References:

<https://archive.ics.uci.edu/ml/datasets/Heart+Disease>