C00528114_Semester-Project

by Hemanth Konda

Submission date: 21-Nov-2023 03:34PM (UTC-0600)
Submission ID: 2234257890
File name: 45872_Hemanth_Konda_C00528114_Semester-Project_1615106_1706068655.pdf (3.57M)
Word count: 4986
Character count: 32667

INFX-502 Systematic Methods in INFX

Semester Project

Heart Attack Analysis & Prediction: Finding the best ML Algorithm in R

University of Louisiana at Lafayette

By

Hemanth Kumar Konda (C00528114)

Instructor

Dr. Mehmet Engin Tozal

Table of Contents

1. Database 1
   1.1. Dataset 1
   1.2. Attributes of the Dataset 1
   1.4. Libraries 2
   1.5. The Dataset 3
   1.6. Descriptive datasets 3
   1.7. Converting the characters into factors 4
   1.8. Cleaning & Pre-Processing the Data 4
      1.8.1. Checking for NULL/Missing Values. 4
      1.8.2. Checking for Duplicates 4
   1.9. Cleaning the Data 5
      1.9.1. Removing Duplicates. 5
   1.10. Statistics of Cleaned Data 5
2. Visualization of variables 8
   2.1. Bar Plots 8
      2.1.1. Bar Plot of Sex 8
      2.1.2. Bar Chart of Target Variable by Age 9
      2.1.3. Bar Plot of Target Variable 10
      2.1.4. Bar Plot of Types of Chest Pain 11
      2.1.5. Bar Plot of Heart Disease by Gender 12
   2.2. Box Plots 13
      2.2.1. Box plot of Age by Sex 13
      2.2.2. Box Plot of Cholesterol Levels by Chest Pain Type 14
      2.2.3. Cholesterol Levels by Gender 15
      2.2.5. Boxplot of Resting Blood Pressure by Heart Attack 16
      2.2.7. Boxplot of Maximum Heart Rate Achieved by Heart Attack Status 16
   2.3. Scatter Plot 17
      2.3.1. SP of Age vs. Resting Blood Pressure 18
      2.3.2. SP of Cholesterol vs. Max Heart Rate 19
      2.3.3. SP of Age vs. Max Heart Rate 20
   2.4. Histograms 21
      2.4.1. Histogram of Numeric features like age, trtbps, chol, thalachh, oldpeak vs Freq. 21

2.4.2. Histogram of Age Distribution 22
2.4.4. Histogram of Serum Cholesterol Level by Heart Disease Status 22
2.4.6. Histogram of Vessels Colored by Fluoroscopy by Heart DiseaseStatus 23
2.4.8. ST Depression by Thalassemia Type 23
2.5. Heatmaps 25
2.5.1. Heatmap: Age vs. Resting Blood Pressure 25
2.6. Density Plots 27
3. Model Development 31
3.1. Pre-Processing 31
3.1.1. Checking for NULL/Missing Values. 31
3.1.2. Checking for Duplicates 31
3.1.3. Removing Duplicates. 32
3.1.4. Statistics of Cleaned Data 32
3.2. Dataset Splitting 33
3.3. K-NN algorithm 34
3.3.1. Confusion matrix of k-NN 36
3.4. Linear Regression 37
3.4. Logistic Regression 39
3.5. NB Classifier 40
3.5.1. Confusion Matrix for Naive Bayes Classifier 41
4. Evaluation 43
4.1. Algorithms Result Table 43
4.2. Future Works 43
4.3. Summary 44

1. Database

1.1. Dataset

The proposed dataset focuses on cardiac studies, which includes 304 rows (example) and 14 columns (features). Rashik Rahman, obtained from Kaggle, was the owner of the dataset. It is optimized for classifying cardiac, complex medical conditions, and each change in the list is carefully detailed in the table provided which serves the purpose of medical stability characteristics have predicted the likelihood of developing heart disease.

1.2. Attributes of the Dataset

Data description of the Variables		
1.	Age	The age of the patient
2.	Sex	The gender of the patient (0 = female, 1 = male).
3.	Cp	Chest pain type (1 = typical angina, 2 = atypical angina, 3 = non-anginal pain, 4 = asymptomatic).
4.	Trestbps	Resting blood pressure (in mm Hg).
5.	Chol	Serum cholesterol level (in mg/dl).
6.	Fbs	Fasting blood sugar (> 120 mg/dl) (1 = true, 0 = false).
7.	Restecg	Resting electrocardiographic results (0 = normal, 1 = having ST-T wave abnormality, 2 = probable or definite left ventricular hypertrophy).
8.	Thalach	Maximum heart rate achieved.
9.	Exang	Exercise-induced angina (1 = yes, 0 = no).
10.	Oldpeak	ST depression induced by exercise relative to rest.
11.	Slope	Slope of the peak exercise ST segment.
12.	Ca	Number of major vessels colored by fluoroscopy.
13.	Thal	Thalassemia (a type of blood disorder) results (3 = normal, 6 = fixed defect, 7 = reversible defect).
14.	Target variable	The target variable in this dataset is often output or target. 0= less chance of heart attack 1= more chance of heart attack

1.3. Expectations

The objective of this initiative is to create and execute predictive algorithms with the purpose of assessing the likelihood of an individual encountering a heart attack or cardiovascular incident in the forthcoming time. Various technologies, including machine learning, data analysis, and medical data, are employed to createaccurate and personalized risk assessment tools. During this undertaking, a set of five machine learning algorithms has been employed, comprising Linear Regression, Logistic Regression, a Naive Bayes model and a KNN Classifier. The goal is to determine which model achieves the highest accuracy in R Programming.

Engaging in medical-related projects such as Heart Attack Prediction & Analysis is crucial because they enable early detection, accurate diagnosis, and personalized treatment.

1.4. Libraries

Prior to importing the dataset, it is essential to initiate the loading of the required libraries.

```
[98] #Loading the Libraries
     library(tidyverse)
     library(ggplot2)

     install.packages("caret")
     library(caret)

     install.packages("plotly")
     library(plotly)
```

```
[98] Installing package into '/usr/local/lib/R/site-library'
     (as 'lib' is unspecified)

     also installing the dependencies 'listenv', 'parallelly', 'future', 'globals', 'shape', 'future.apply', 'numDeriv', 'progressr', 'SQUAREM', 'diagram', 'lava', 'prodlim', 'proxy', 'iterators', 'clock', 'gower', 'hardhat', 'ipred', 'sfd

     Loading required package: lattice

     Attaching package: 'caret'

     The following object is masked from 'package:purrr':

         lift

     Installing package into '/usr/local/lib/R/site-library'
     (as 'lib' is unspecified)

     also installing the dependencies 'later', 'htmlwidgets', 'lazyeval', 'crosstalk', 'promises'


     Attaching package: 'plotly'

     The following object is masked from 'package:ggplot2':

         last_plot

     The following object is masked from 'package:stats':

         filter

     The following object is masked from 'package:graphics':

         layout
```

The **Tidyverse** library contains a collection of R packages, such as **ggplot2**, **dplyr**, and **tidyr**, designed to jointly simplify the tasks of observational data manipulation, visualization, and analysis In the R programming, the **ggplot2** library stands out as a robust and flexible plotting system to users. In addition to the ability to generate multiple visual representations, the **caret** package plays an important role in machine learning, supporting the tasks of pattern training, analysis, and of forecasting, and provides a unified interface for models for different mechanisms.

1.5. The Dataset

I have named my dataset 'heart' and saved it in Microsoft Excel in CSV format, i.e., '**heart.csv**'.

```
[ ]  # Reading CSV file
     df <- read.csv("heart.csv")
```

1.6. Descriptive datasets

The code below indicates that it will display the data in the dataset. It specifies that the dataset comprises 303 columns and 14 rows.

```
[ ]  # Displaying information about the dataframe
     str(df)
```

```
'data.frame':   303 obs. of  14 variables:
 $ age     : int  63 37 41 56 57 57 56 44 52 57 ...
 $ sex     : chr  "Male" "Male" "Female" "Male" ...
 $ cp      : chr  "Asymptomatic" "Non-Anginal Pain" "Atypical Angina" "Atypical Angina" ...
 $ trtbps  : int  145 130 130 120 120 140 140 120 172 150 ...
 $ chol    : int  233 250 204 236 354 192 294 263 199 168 ...
 $ fbs     : int  1 0 0 0 0 0 0 0 1 0 ...
 $ restecg : int  0 1 0 1 1 1 0 1 1 1 ...
 $ thalachh: int  150 187 172 178 163 148 153 173 162 174 ...
 $ exng    : int  0 0 0 0 1 0 0 0 0 0 ...
 $ oldpeak : num  2.3 3.5 1.4 0.8 0.6 0.4 1.3 0 0.5 1.6 ...
 $ slp     : int  0 0 2 2 2 1 1 2 2 2 ...
 $ caa     : int  0 0 0 0 0 0 0 0 0 0 ...
 $ thall   : int  1 2 2 2 2 1 2 3 3 2 ...
 $ output  : int  1 1 1 1 1 1 1 1 1 1 ...
```

1.7. Converting the characters into factors

The code below demonstrates the conversion of character-based attributes, such as 'sex' (gender of the patient), 'cp' (chest pain), 'restecg' (resting electrocardiographic results), 'exng' (exercise-induced angina), and 'thall' (maximum heart rate achieved), into factors.

```
[6]  df$sex = factor(df$sex)
     df$cp = factor(df$cp)
     df$restecg = factor(df$restecg)
     df$exng = factor(df$exng)
     df$thall = factor(df$thall)
```

1.8. Cleaning & Pre-Processing the Data

1.8.1. Checking for NULL/Missing Values.

The first step in data pre-processing is checking for NULL/missing values in the dataset. The code below checks for NULL values in the dataset. The output indicates that the dataset doesn't have any NULL/missing values in any of the attributes/columns.

```
[8]  # Checking for missing values
     print(colSums(is.na(df)))
```

```
      age       sex        cp    trtbps      chol       fbs   restecg  thalachh
        0         0         0         0         0         0         0         0
    exng   oldpeak       slp       caa     thall    output
        0         0         0         0         0         0
```

1.8.2. Checking for Duplicates

The second step in data pre-processing is checking for duplicates in the dataset. The code below verifies whether any duplicates exist in the dataset. According to the output, there is one duplicate value in the dataset.

```
[8]  # Checking for duplicated rows
     print(sum(duplicated(df)))
```

```
[1] 1
```

The code below identifies and displays duplicate rows in the dataset. According to the output, row 165 is a duplicate of row 166 in the dataset.

```
[9]  # Displaying duplicated rows
     print(df[duplicated(df), ])
```

```
     age  sex              cp trtbps chol   fbs  restecg thalachh exng oldpeak
165   38 Male Non-Anginal Pain    138  175 FALSE STT Wave      173   No       0
     slp caa        thall output
165    2   4 Fixed Defect      1
```

1.9. Cleaning the Data

1.9.1. Removing Duplicates.

The code below is used to remove duplicate rows in the dataset.

```
[10] # Removing duplicated rows
     df <- df[!duplicated(df), ]
```

The code below is used to display the updated dataset.

```
[11] # Displaying the updated dataframe shape
     print(dim(df))

     [1] 302  14
```

Now the dataset only has 302 rows; previously, it had 303 rows.

1.10. Statistics of Cleaned Data

After cleaning the dataset, observe the dataset. The code below is used to display a summary of the dataset.

```
[15] # Displaying summary statistics
     print(summary(df))
```

```
      age            sex                      cp          trtbps      
 Min.   :29.00   Female: 96   Asymptomatic    : 23   Min.   : 94.0  
 1st Qu.:48.00   Male  :206   Atypical Angina : 50   1st Qu.:120.0  
 Median :55.50                Non-Anginal Pain: 86   Median :130.0  
 Mean   :54.42                Typical Angina  :143   Mean   :131.6  
 3rd Qu.:61.00                                       3rd Qu.:140.0  
 Max.   :77.00                                       Max.   :200.0  
      chol          fbs             restecg       thalachh      exng       
 Min.   :126.0   Mode :logical   LVH     :  4   Min.   : 71.0   No :203  
 1st Qu.:211.0   FALSE:257       Normal  :147   1st Qu.:133.2   Yes: 99  
 Median :240.5   TRUE :45        STT Wave:151   Median :152.5            
 Mean   :246.5                                  Mean   :149.6            
 3rd Qu.:274.8                                  3rd Qu.:166.0            
 Max.   :564.0                                  Max.   :202.0            
    oldpeak           slp              caa                      thall     
 Min.   :0.000   Min.   :0.000   Min.   :0.0000   Fixed Defect     :165  
 1st Qu.:0.000   1st Qu.:1.000   1st Qu.:0.0000   Normal           : 20  
 Median :0.800   Median :1.000   Median :0.0000   Reversable Defect:117  
 Mean   :1.043   Mean   :1.397   Mean   :0.7185                          
 3rd Qu.:1.600   3rd Qu.:2.000   3rd Qu.:1.0000                          
 Max.   :6.200   Max.   :2.000   Max.   :4.0000                          
     output     
 Min.   :0.000  
 1st Qu.:0.000  
 Median :1.000  
 Mean   :0.543  
 3rd Qu.:1.000  
 Max.   :1.000  
```

```
[13] str(df)
```

```
'data.frame':   302 obs. of  14 variables:
 $ age     : int  63 37 41 56 57 57 56 44 52 57 ...
 $ sex     : Factor w/ 2 levels "Female","Male": 2 2 1 2 1 2 1 2 2 2 ...
 $ cp      : Factor w/ 4 levels "Asymptomatic",..: 1 3 2 2 4 4 2 2 3 3 ...
 $ trtbps  : int  145 130 130 120 120 140 140 120 172 150 ...
 $ chol    : int  233 250 204 236 354 192 294 263 199 168 ...
 $ fbs     : logi  TRUE FALSE FALSE FALSE FALSE FALSE ...
 $ restecg : Factor w/ 3 levels "LVH","Normal",..: 2 3 2 3 3 3 2 3 3 3 ...
 $ thalachh: int  150 187 172 178 163 148 153 173 162 174 ...
 $ exng    : Factor w/ 2 levels "No","Yes": 1 1 1 1 2 1 1 1 1 1 ...
 $ oldpeak : num  2.3 3.5 1.4 0.8 0.6 0.4 1.3 0 0.5 1.6 ...
 $ slp     : int  0 0 2 2 2 1 1 2 2 2 ...
 $ caa     : int  0 0 0 0 0 0 0 0 0 0 ...
 $ thall   : Factor w/ 3 levels "Fixed Defect",..: 2 1 1 1 1 2 1 3 3 1 ...
 $ output  : int  1 1 1 1 1 1 1 1 1 1 ...
```

The following code is used to display the head of the dataset.

```
[15] head(df)
```

A data.frame: 6 × 14

	age	sex	cp	trtbps	chol	fbs	restecg	thalachh	exng	oldpeak	slp	caa	thall	output
	<int>	<fct>	<fct>	<int>	<int>	<lgl>	<fct>	<int>	<fct>	<dbl>	<int>	<int>	<fct>	<int>
1	63	Male	Asymptomatic	145	233	TRUE	Normal	150	No	2.3	0	0	Normal	1
2	37	Male	Non-Anginal Pain	130	250	FALSE	STT Wave	187	No	3.5	0	0	Fixed Defect	1
3	41	Female	Atypical Angina	130	204	FALSE	Normal	172	No	1.4	2	0	Fixed Defect	1
4	56	Male	Atypical Angina	120	236	FALSE	STT Wave	178	No	0.8	2	0	Fixed Defect	1
5	57	Female	Typical Angina	120	354	FALSE	STT Wave	163	Yes	0.6	2	0	Fixed Defect	1
6	57	Male	Typical Angina	140	192	FALSE	STT Wave	148	No	0.4	1	0	Normal	1

The code below is used to display the tail of the dataset.

```
[14] tail(df)
```

A data.frame: 6 × 14

	age	sex	cp	trtbps	chol	fbs	restecg	thalachh	exng	oldpeak	slp	caa	thall	output
	<int>	<fct>	<fct>	<int>	<int>	<lgl>	<fct>	<int>	<fct>	<dbl>	<int>	<int>	<fct>	<int>
298	59	Male	Typical Angina	164	176	TRUE	Normal	90	No	1.0	1	2	Normal	0
299	57	Female	Typical Angina	140	241	FALSE	STT Wave	123	Yes	0.2	1	0	Reversable Defect	0
300	45	Male	Asymptomatic	110	264	FALSE	STT Wave	132	No	1.2	1	0	Reversable Defect	0
301	68	Male	Typical Angina	144	193	TRUE	STT Wave	141	No	3.4	1	2	Reversable Defect	0
302	57	Male	Typical Angina	130	131	FALSE	STT Wave	115	Yes	1.2	1	1	Reversable Defect	0
303	57	Female	Atypical Angina	130	236	FALSE	Normal	174	No	0.0	1	1	Fixed Defect	0

2. Visualization of variables

To gain a comprehensive understanding of the dataset and the relationships between various variables, I have generated numerous plots depicting different variables.

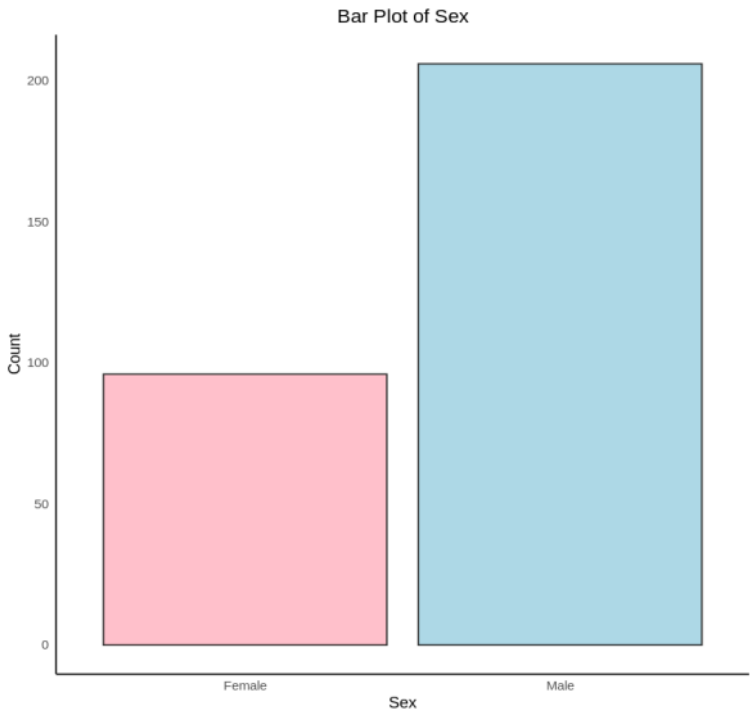
2.1. Bar Plots

A graphical depiction known as a box plot illustrates the connection between a numerical and a categorical variable within a dataset. In this representation, each category is depicted as a bar, the length of which corresponds to its numeric value.

To better understand the dataset's complexity, I developed a number of bar plots, each of which provides a unique viewpoint on the data.

2.1.1. Bar Plot of Sex.

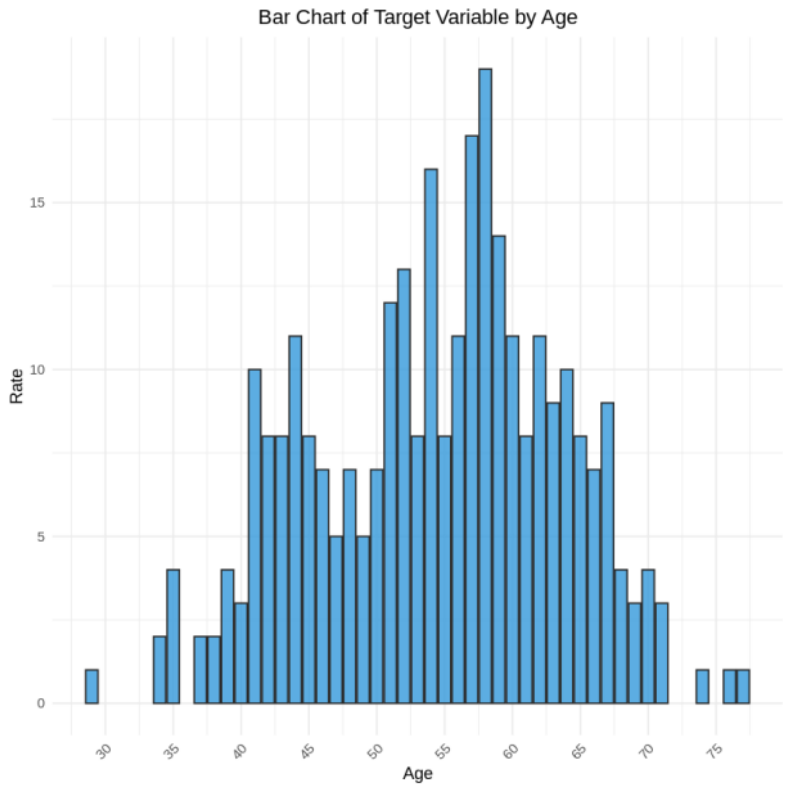
```
[ ]  ggplot(df, aes(x = sex)) +
       geom_bar(fill = c("pink", "lightblue"), color = "#2b2b2b") +
       labs(title = "Bar Plot of Sex", y = "Count", x = "Sex") +
       theme_minimal() +
       theme(
         plot.title = element_text(hjust = 0.5),  # Centering the title
         panel.grid.major = element_blank(),  # Removing major grid lines
         panel.grid.minor = element_blank(),  # Removing minor grid lines
         panel.border = element_blank(),  # Removing the panel border
         axis.line = element_line(color = "black")  # Adding axis line
       )
```



According to this bar plot, we can understand that the dataset has a greater number of male patients than female patients. The count of male patients is 200, while the count of female patients is around 100.

2.1.2. Bar Chart of Target Variable by Age

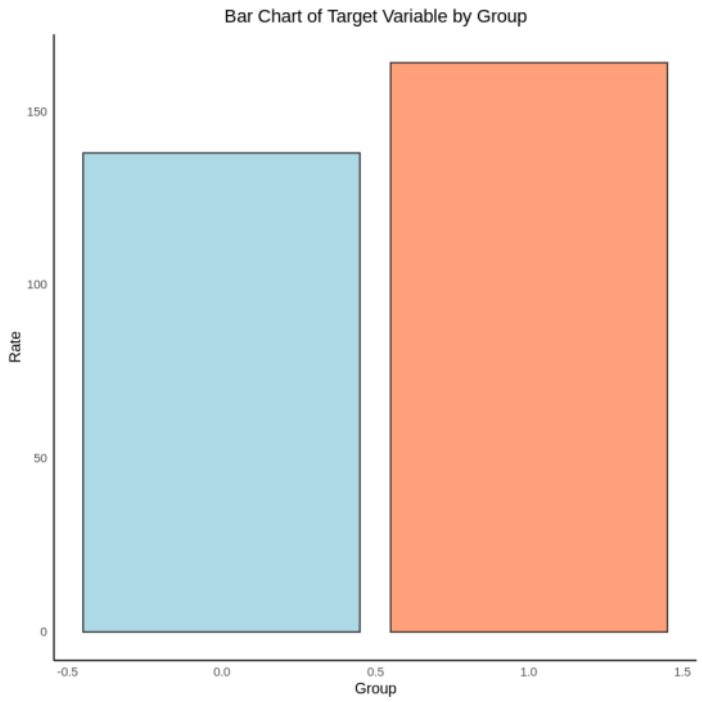
```
[99] ggplot(df, aes(x = age)) +
       geom_bar(fill = "#3498db", color = "#2b2b2b", alpha = 0.8) +  # Customizing bar chart colors
       labs(title = "Bar Plot of Target Variable by Age", y = "Rate", x = "Age") +  # Adding labels
       theme_minimal() +
       theme(
         plot.title = element_text(hjust = 0.5),  # Centering the title
         axis.text.x = element_text(angle = 45, hjust = 1)  # Rotating x-axis labels for better readability
       ) +
       scale_x_continuous(breaks = seq(25, 90, by = 5))
```



According to this plot, we can understand that there are more patients aged 58, falling within the 18-20 age group. The second-highest age group is 57 years old, comprising 12 individuals.

2.1.3. Bar Plot of Target Variable

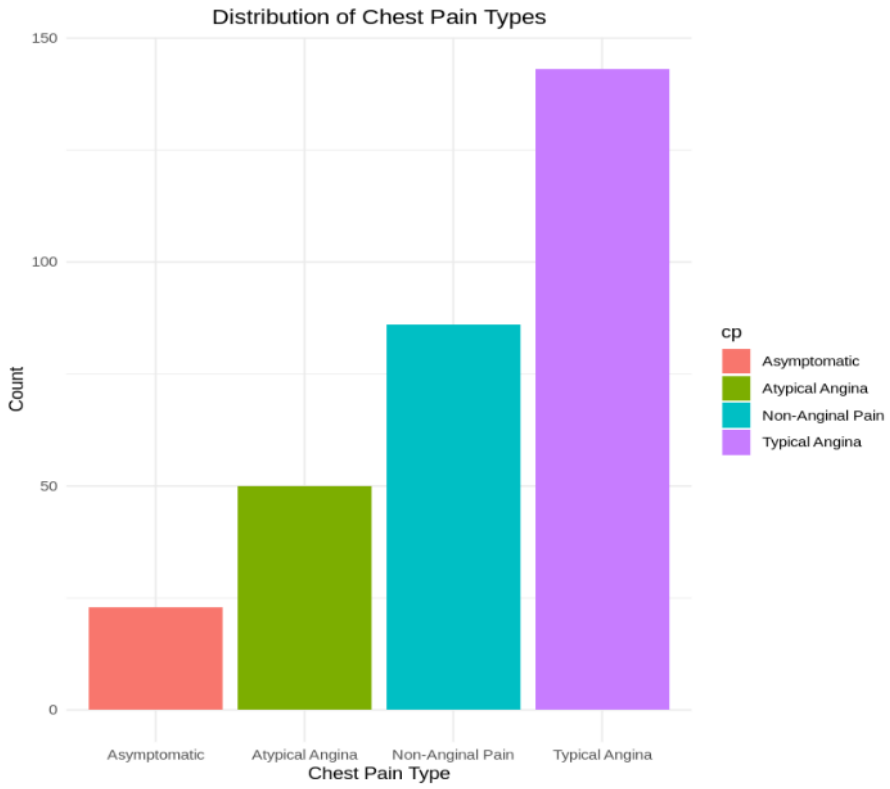
```
ggplot(df, aes(x = output)) +
  geom_bar(fill = c("#ADD8E6", "#FFA07A"), color = "#2b2b2b") +  # Light blue and light red colors
  labs(title = "Bar Plot of Target Variable by Group", y = "Rate", x = "Group") +  # Adding labels
  theme_minimal() +
  theme(
    plot.title = element_text(hjust = 0.5),  # Centering the title
    panel.grid.major = element_blank(),  # Removing major grid lines
    panel.grid.minor = element_blank(),  # Removing minor grid lines
    panel.border = element_blank(),  # Removing the panel border
    axis.line = element_line(color = "black")  # Adding axis line
  )
```



Examination of the bar chart indicates that the probability of encountering a heart attack (1) surpasses the probability of not encountering a heart attack (0). The incidence rate of heart attacks exceeds 150, whereas the probability of not having a heart attack falls within the range of 145- 150.

2.1.4. Bar Plot of Types of Chest Pain

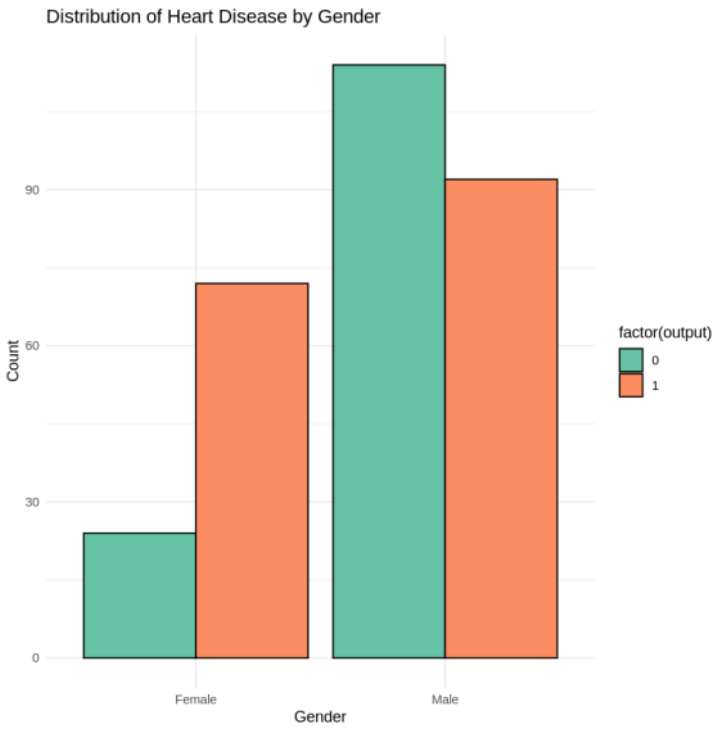
```
[ ]  ggplot(df, aes(x = cp, fill = cp)) +
       geom_bar() +
       labs(title = "Distribution of Chest Pain Types", y = "Count", x = "Chest Pain Type") +
       theme_minimal() +
       theme(
         plot.title = element_text(hjust = 0.5)  # Centering the title
       )
```



The chart illustrates four categories of chest discomfort. The most frequently occurring type is Type 1, typical angina, with a count of around 145, while the least common is Type 4, asymptomatic, which occurs approximately 29 times.

2.1.5. Bar Plot of Heart Disease by Gender

```
# Create a Bar Plot
ggplot(df, aes(x = sex, fill = factor(output))) +
  geom_bar(position = "dodge", color = "black") +
  labs(title = "Distribution of Heart Disease by Gender", x = "Gender", y = "Count") +
  scale_fill_manual(values = c("#66c2a5", "#fc8d62")) +  # Custom colors for each output level
  theme_minimal()
```



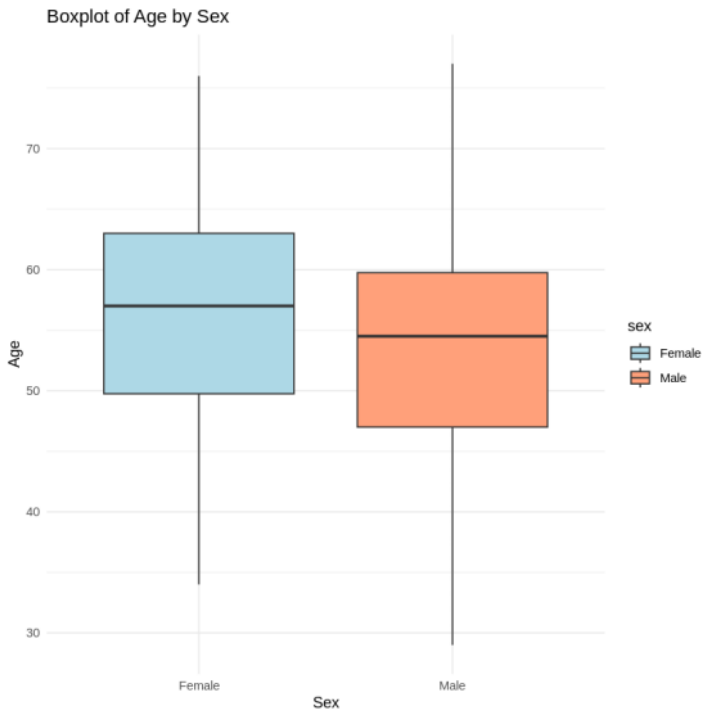
From this plot, we can understand that heart disease is more common in male patients than in females.

2.2. Box Plots

Box plot serves to summarize a dataset. This visual depiction effectively illustrates the distribution of data, highlighting outliers and conveying a rapid overview of value variability within the dataset. The box plot provides insight into essential statistical parameters, including the median, upper and lower quartiles, and the range between minimum and maximum values. This visualization helps identify potential anomalies or inaccuracies within the data.

2.2.1. Box plot of Age by Sex

```
[ ]  ggplot(df, aes(x = as.factor(sex), y = age, fill = sex)) +
       geom_boxplot() +
       labs(title = "Boxplot of Age by Sex", y = "Age", x = "Sex") +
       scale_fill_manual(values = c("#ADD8E6", "#FFA07A")) +  # Light blue for male, light red for female
       theme_minimal()
```

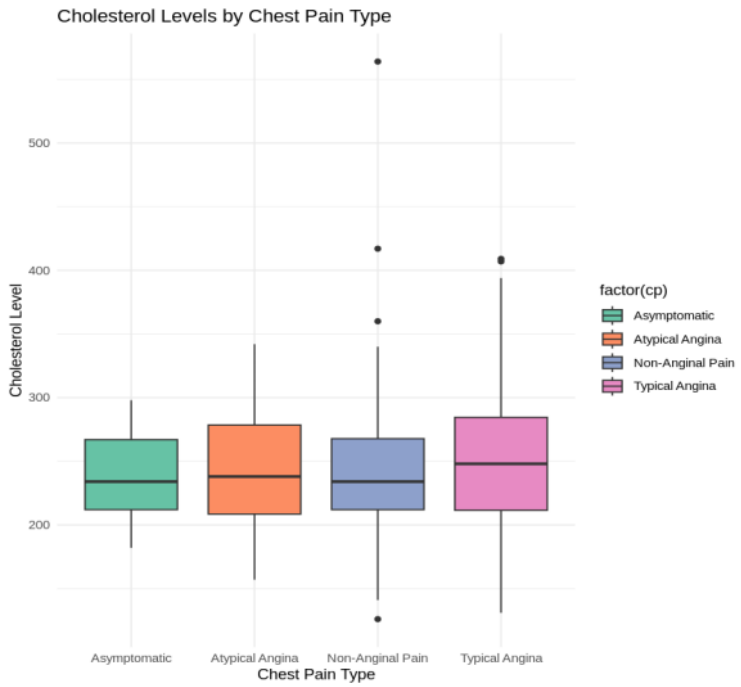


Based on the depicted box plot, it becomes apparent that the age range for female patients spans from 34 to 71 years, while male patients exhibit an age range of 29 to 72 years. The median age for females is 57 years, in contrast to males whose median age is 54 years.

2.2.2. Box Plot of Cholesterol Levels by Chest Pain Type

```
[ ]  # Load necessary libraries
     library(ggplot2)

     # Create a boxplot
     ggplot(df, aes(x = factor(cp), y = chol, fill = factor(cp))) +
       geom_boxplot() +
       labs(title = "Cholesterol Levels by Chest Pain Type", y = "Cholesterol Level", x = "Chest Pain Type") +
       scale_fill_manual(values = c("#66c2a5", "#fc8d62", "#8da0cb", "#e78ac3")) +  # Custom colors for each chest pain type
       theme_minimal()
```

From the box plot below, we can understand that Typical Angina is the most common, and there are some outliers in non-Anginal; there are more outliers.

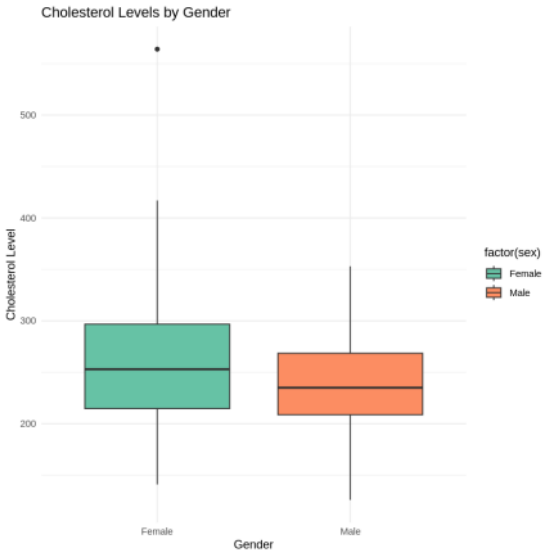


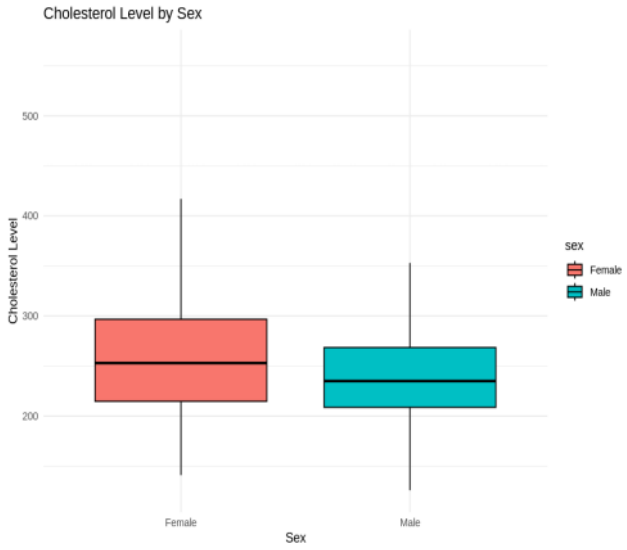
2.2.3. Cholesterol Levels by Gender

```
# Create a boxplot
ggplot(df, aes(x = factor(sex), y = chol, fill = factor(sex))) +
  geom_boxplot() +
  labs(title = "Cholesterol Levels by Gender", y = "Cholesterol Level", x = "Gender") +
  scale_fill_manual(values = c("#66c2a5", "#fc8d62")) +  # Custom colors for each gender
  theme_minimal()
```

2.2.4. Box Plot of Cholesterol Level by Sex

```
[ ] # Assuming df is your data frame
    library(ggplot2)

    # Create a boxplot
    ggplot(df, aes(x = sex, y = chol, fill = sex)) +
      geom_boxplot(color = "black", outlier.shape = NA) +  # Hide outliers for better visibility
      labs(title = "Cholesterol Level by Sex", x = "Sex", y = "Cholesterol Level") +
      theme_minimal()
```





2.2.5. Boxplot of Resting Blood Pressure by Heart Attack

```
#Resting Blood Pressure (Trestbps) Box Plot
ggplot(df, aes(x = "", y = trtbps, fill = factor(output))) +
  geom_boxplot() +
  labs(title = "Boxplot of Resting Blood Pressure by Heart Attack Status", x = "", y = "Resting Blood Pressure") +
  theme_minimal() +
  scale_fill_manual(values = c("0" = "lightcoral", "1" = "lightskyblue"))
```

2.2.6. Boxplot of Serum Cholesterol Level by Heart Attack Status

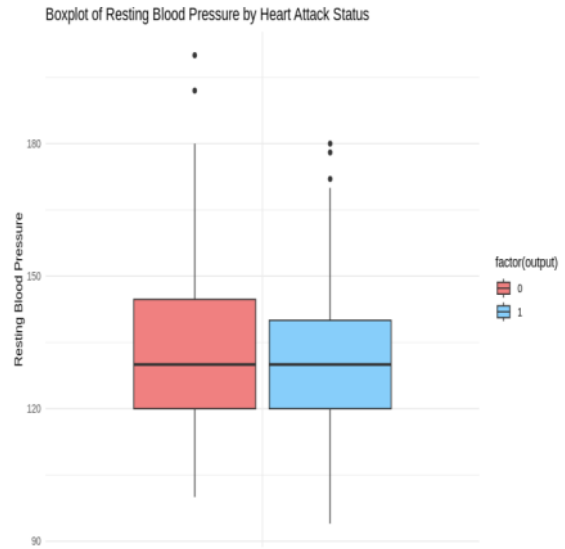
```
#Serum Cholesterol Level (Chol) Box Plot
ggplot(df, aes(x = "", y = chol, fill = factor(output))) +
  geom_boxplot() +
  labs(title = "Boxplot of Serum Cholesterol Level by Heart Attack Status", x = "", y = "Serum Cholesterol Level") +
  theme_minimal() +
  scale_fill_manual(values = c("0" = "lightcoral", "1" = "lightskyblue"))
```

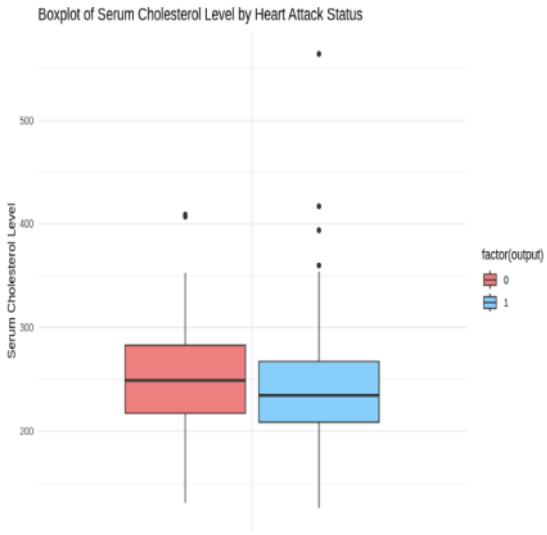
2.2.7. Boxplot of Maximum Heart Rate Achieved by Heart Attack Status

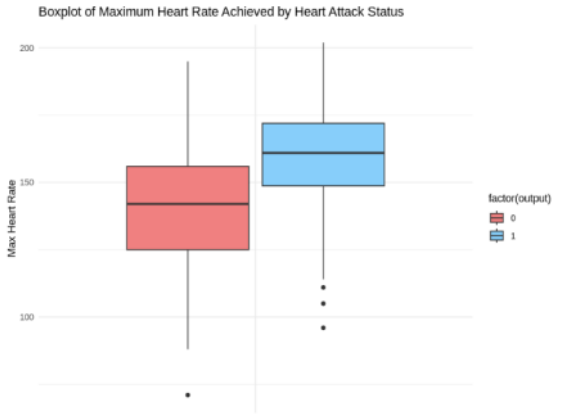
```
#Maximum Heart Rate Achieved (Thalach) Box Plot
ggplot(df, aes(x = "", y = thalachh, fill = factor(output))) +
  geom_boxplot() +
  labs(title = "Boxplot of Maximum Heart Rate Achieved by Heart Attack Status", x = "", y = "Max Heart Rate") +
  theme_minimal() +
  scale_fill_manual(values = c("0" = "lightcoral", "1" = "lightskyblue"))
```

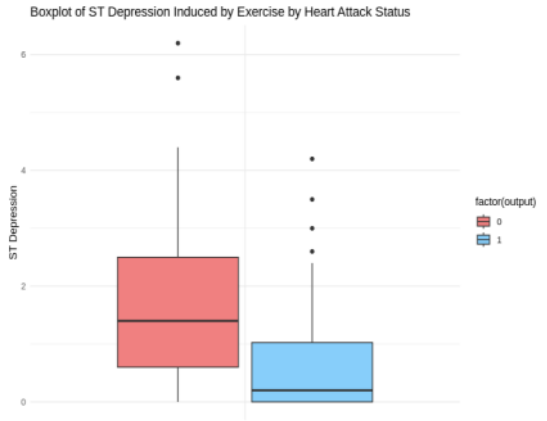
2.2.8. Boxplot of ST Depression Induced by Exercise by Heart Attack Status

```
[ ] #ST Depression Induced by Exercise (Oldpeak) Box Plot
    ggplot(df, aes(x = "", y = oldpeak, fill = factor(output))) +
      geom_boxplot() +
      labs(title = "Boxplot of ST Depression Induced by Exercise by Heart Attack Status", x = "", y = "ST Depression") +
      theme_minimal() +
      scale_fill_manual(values = c("0" = "lightcoral", "1" = "lightskyblue"))
```









2.3. Scatter Plot

The positioning of each dot on both the x and y axes indicates the values associated with a particular data point. These graphs are employed to analyze relationships between variables.

These scatter plots we can create based on the dataset.

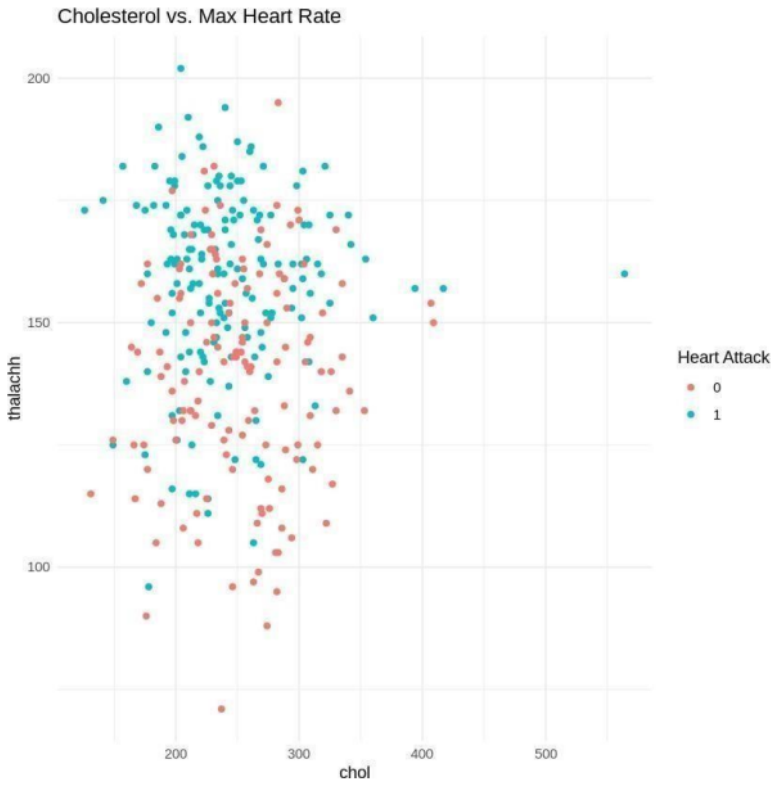
2.3.1. SP of Age vs. Resting Blood Pressure

```
[ ]  #Age vs. Resting Blood Pressure (Trtbps):
     ggplot(df, aes(x = age, y = trtbps, color = factor(output))) +
       geom_point() +
       labs(title = "Age vs. Resting Blood Pressure", color = "Heart Attack") +
       theme_minimal()
```

In the provided scatter diagram, x correspond to the 'age' of the subjects, while y corresponds to 'trtbps' (Resting blood pressure, measured in mm Hg). It is noticeable that the probability of encountering a heart attack tends to increase in the age range of 40-70, particularly when the resting blood pressure falls within the range of 120 mm Hg to 150 mm Hg.

2.3.2. SP of Cholesterol vs. Max Heart Rate

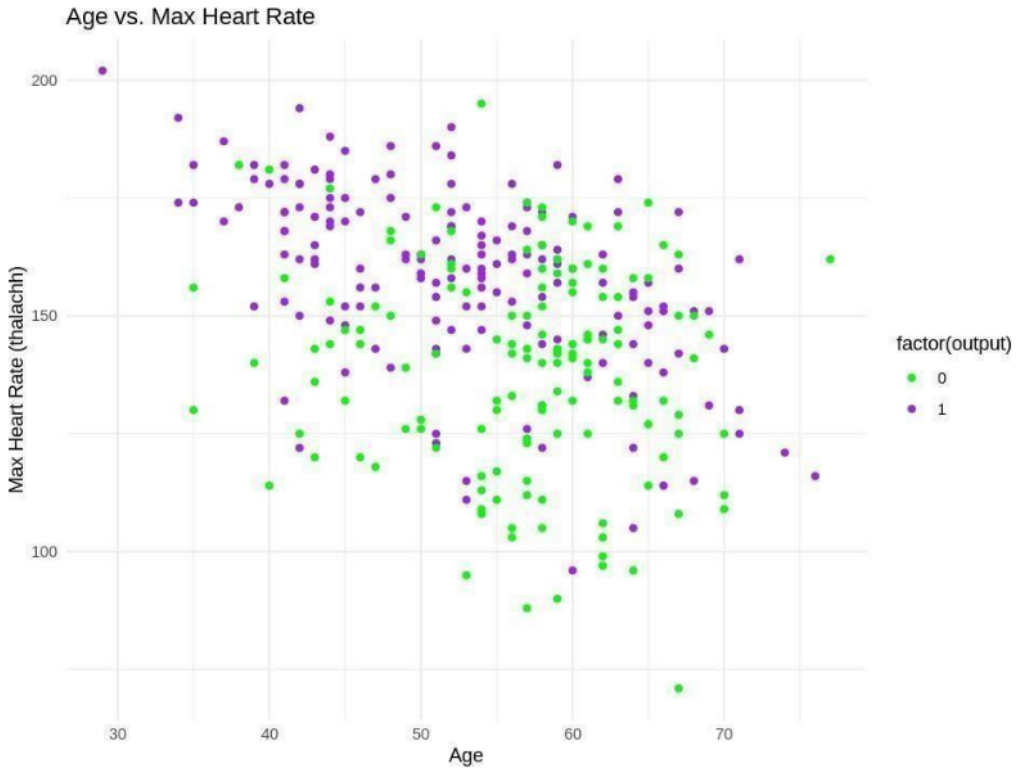
```
#Serum Cholesterol Level (Chol) vs. Maximum Heart Rate (Thalachh)
ggplot(df, aes(x = chol, y = thalachh, color = factor(output))) +
  geom_point() +
  labs(title = "Cholesterol vs. Max Heart Rate", color = "Heart Attack") +
  theme_minimal()
```



In this scatter plot, 'chol' (Serum cholesterol level in mg/dl) is on the x-axis, and 'thalachh' (Maximum heart rate achieved) is on the y-axis. A score of 0 signifies a reduced probability of a heart attack, whereas a score of 1 suggests an elevated likelihood. It is evident that individuals with a 'chol' level ranging from 200 mg to 300 mg, coupled with a peak heart rate falling within the 150 to 200 range, are more prone to experiencing a heart attack.

2.3.3. SP of Age vs. Max Heart Rate

```
[ ]  # Set the size of the plot
     options(repr.plot.width=8, repr.plot.height=6)

     # Create the scatter plot
     ggplot(df, aes(x = age, y = thalachh, color = factor(output))) +
       geom_point() +
       labs(x = 'Age', y = 'Max Heart Rate (thalachh)', title = 'Age vs. Max Heart Rate') +
       scale_color_manual(values = c("0" = "green", "1" = "purple")) +
       theme_minimal()
```



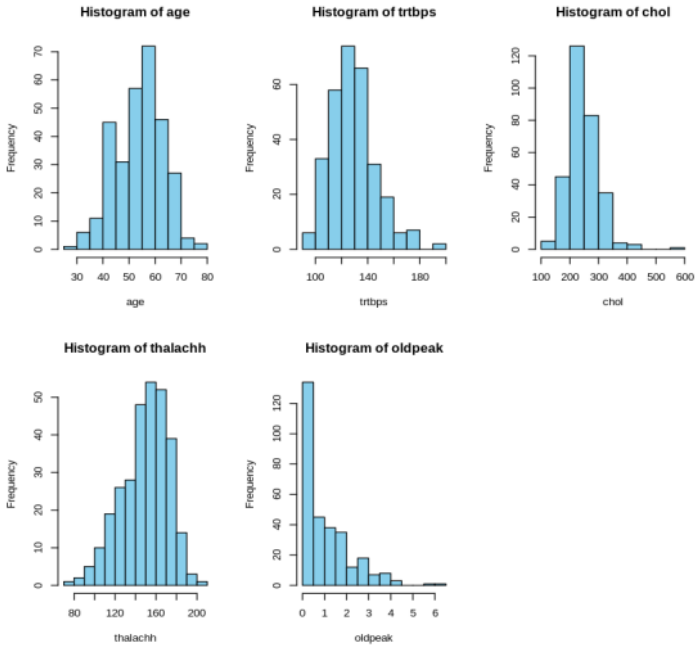
In this scatter plot, the 'Age' of the patients is on the x-axis, and 'thalachh' (Maximum heart rate achieved) is on the y-axis. A score of 0 signifies a reduced likelihood of a cardiac event, whereas a score of 1 suggests an elevated risk. It is noticeable that the probability of encountering a heart attack is increased in individuals within the age range of 35-60, with the peak heart rate achieved falling between 130 and 200.

2.4. Histograms

A histogram serves as a visual depiction resembling a bar graph, illustrating data by categorizing a set of classes into vertical columns along X-axis. Y-axis, positioned vertically, indicates count or percentage of occurrences corresponding to each column in the data. These columns offer a means to visually interpret the patterns present in the distribution of data. This graphical tool proves valuable in summarizing information related to discrete or continuous data measured on an interval scale. These are the histograms we can create based on the dataset

2.4.1. Histogram of Numeric features like age, trtbps, chol, thalachh, oldpeak vs Freq.

```
[ ]  # Numeric features
     numeric_features <- c('age', 'trtbps', 'chol', 'thalachh', 'oldpeak')

     # Create a histogram
     par(mfrow=c(2, 3))  # Setting up a 2x3 grid for subplots
     for (feature in numeric_features) {
       hist(df[[feature]], main = paste("Histogram of", feature), xlab = feature, col = "#87CEEB", border = "black", breaks = 15)
     }
```



The depicted bar charts illustrate frequencies, with the y-axis indicating occurrence rates and the x-axis featuring different factors like age, trtbps, chol, thalachh, and oldpeak. The frequency is indicative of the probability of experiencing a heart attack—higher frequencies correspond to an increased likelihood of a heart attack, while lower frequencies signify a reduced probability of a heart attack.

The first plot determines that there is more chance in patients aged between 50 and 65. The second plot determines that there is a more chance in patients who have a resting blood pressure of 125 mm Hg. The third plot determines that there is a higher chance of a heart attack in patients who have a serum cholesterol level of 250 mg. The fourth plot determines that there is a higher chance of a heart attack in patients who have a maximum heart rate of 135-160.

2.4.2. Histogram of Age Distribution

```
# Create a histogram with ggplot2
ggplot(df, aes(x = age, fill = ..count..)) +
  geom_histogram(binwidth = 5, color = 'white', position = 'identity') +
  labs(title = 'Age Distribution', x = 'Age', y = 'Frequency') +
  theme_minimal()
```

2.4.3. Histogram of Resting Blood Pressure by Chest Pain

```
[ ] #Resting Blood Pressure Distribution by Chest Pain Type
ggplot(df, aes(x = trtbps, fill = factor(cp))) +
  geom_histogram(position = "dodge", bins = 20, color = 'white') +
  labs(title = 'Resting Blood Pressure Distribution by Chest Pain Type', x = 'Resting Blood Pressure', y = 'Frequency', fill = 'Chest Pain Type') +
  theme_minimal()
```

2.4.4. Histogram of Serum Cholesterol Level by Heart Disease Status

```
#Serum Cholesterol Level by Heart Disease Status
ggplot(df, aes(x = chol, fill = factor(output))) +
  geom_histogram(position = "dodge", bins = 20, color = 'white') +
  labs(title = 'Serum Cholesterol Level by Heart Disease Status', x = 'Serum Cholesterol Level', y = 'Frequency', fill = 'Heart Disease Status') +
  theme_minimal()
```

2.4.5. Histogram of MHR Achieved by EIA

```
[ ] #Maximum Heart Rate Achieved by Exercise-Induced Angina
ggplot(df, aes(x = thalachh, fill = factor(exng))) +
  geom_histogram(position = "dodge", bins = 20, color = 'white') +
  labs(title = 'Maximum Heart Rate Achieved by Exercise-Induced Angina', x = 'Maximum Heart Rate Achieved', y = 'Frequency', fill = 'Exercise-Induced
  theme_minimal()
```

2.4.6. Histogram of Vessels Colored by Fluoroscopy by Heart DiseaseStatus

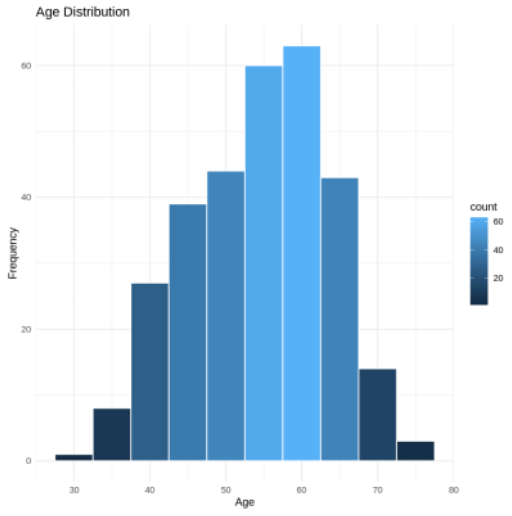
```
[ ] #Number of Major Vessels Colored by Fluoroscopy by Heart Disease Status
    ggplot(df, aes(x = caa, fill = factor(output))) +
      geom_histogram(position = "dodge", bins = 20, color = 'white') +
      labs(title = 'Number of Major Vessels Colored by Fluoroscopy by Heart Disease Status', x = 'Number of Vessels', y = 'Frequency', fill = 'Heart Disea
      theme_minimal()
```

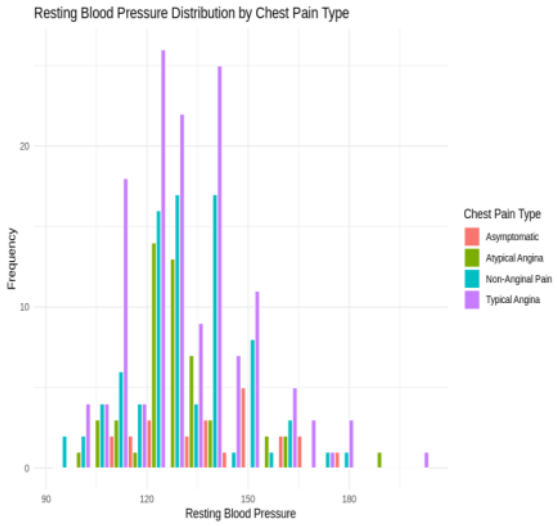
2.4.7. Age Distribution by Gender

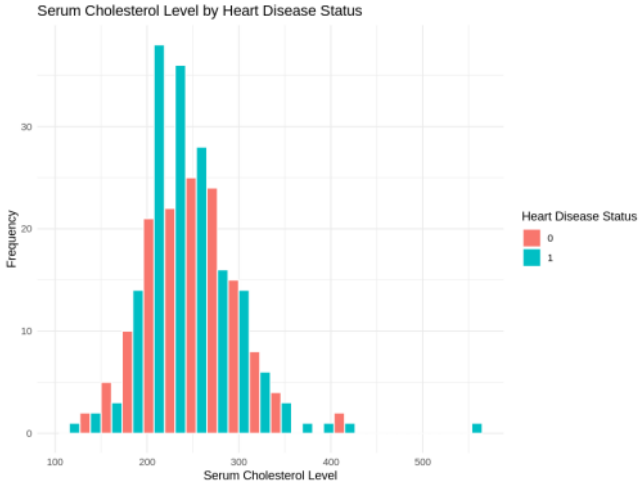
```
#Age Distribution by Gender
ggplot(df, aes(x = age, fill = factor(sex))) +
  geom_histogram(position = "dodge", bins = 20, color = 'white') +
  labs(title = 'Age Distribution by Gender', x = 'Age', y = 'Frequency', fill = 'Gender') +
  theme_minimal()
```

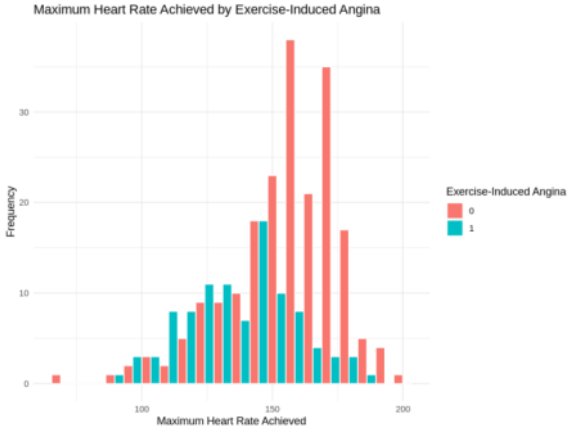
2.4.8. ST Depression by Thalassemia Type

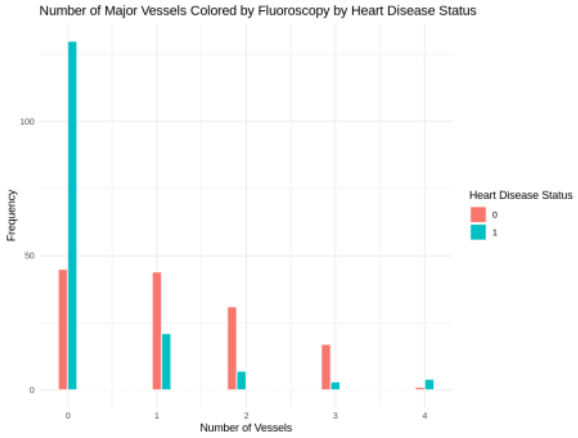
```
[ ] #ST Depression by Thalassemia Type
    ggplot(df, aes(x = oldpeak, fill = factor(thall))) +
      geom_histogram(position = "dodge", bins = 20, color = 'white') +
      labs(title = 'ST Depression by Thalassemia Type', x = 'ST Depression', y = 'Frequency', fill = 'Thalassemia Type') +
      theme_minimal()
```



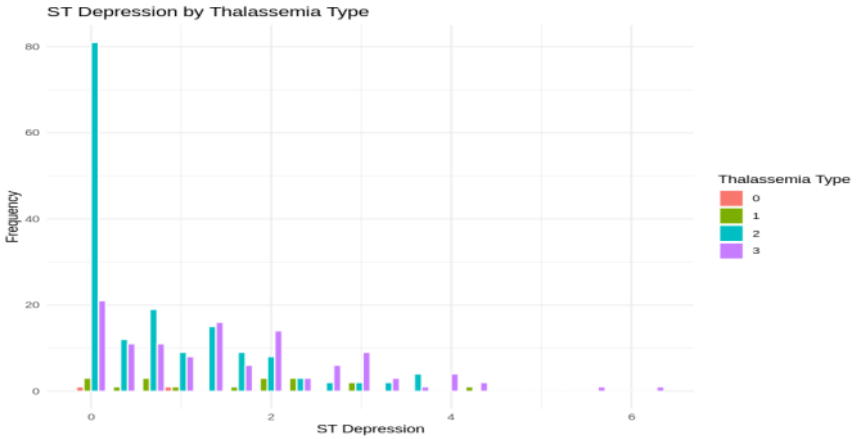



Serum Cholesterol Level by Heart Disease Status
Frequency
Serum Cholesterol Level
Heart Disease Status
0
1


Maximum Heart Rate Achieved by Exercise-Induced Angina
Frequency
Maximum Heart Rate Achieved
Exercise-Induced Angina
0
1


Number of Major Vessels Colored by Fluoroscopy by Heart Disease Status
Frequency
Number of Vessels
Heart Disease Status
0
1


Age Distribution by Gender
Frequency
Age
Gender
Female
Male


ST Depression by Thalassemia Type
Frequency
ST Depression
Thalassemia Type
0
1
2
3

- The age distribution histogram shows that patients aged between 55 and 60 have a higher frequency and likelihood of getting a heart attack.
- The Histogram of Resting BP Distribution by Chest Pain Type indicates that Typical Angina has a higher likelihood of causing a heart attack with a Resting Blood Pressure of 120 - 150 mm Hg.
- Histogram of Cholesterol by Heart Disease states that there is a higher likelihood of getting a heart attack with cholesterol levels between 200 and 300.
- The histogram of Age Distribution by Gender states that male patients have a higher likelihood of getting a heart attack between the ages of 40 and 60.

2.5. Heatmaps

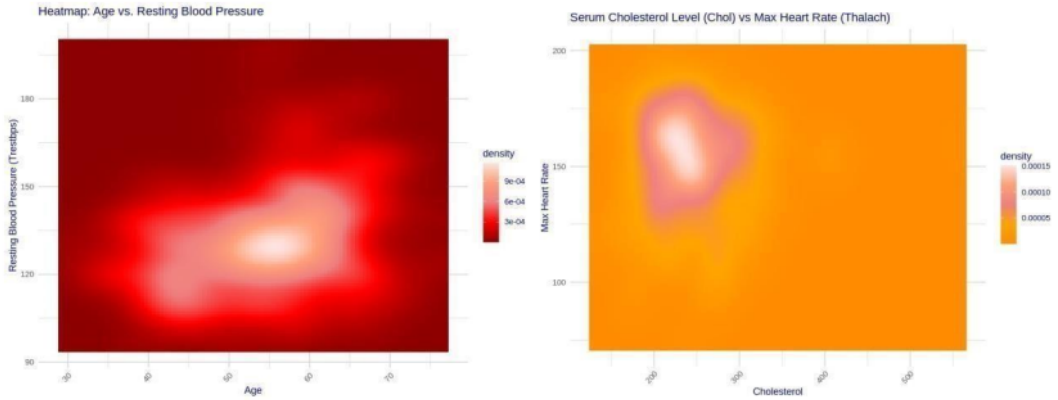
A heat map serves as a visual representation of data in two dimensions, employing a color scheme to depict the intensity or magnitude of individual values within a given dataset. The color gradation can be determined by either hue or intensity. The main objective of utilizing heat maps is to improve the clarity of presenting the distribution of locations or events within the dataset, guiding observers to the most significant areas on visualizations.

2.5.1. Heatmap: Age vs. Resting Blood Pressure

```
[ ]  color_theme <- colorRampPalette(c('darkred', 'red', 'lightcoral', 'lightsalmon', 'mistyrose'))(50)

     # Heatmap for Age and Resting Blood Pressure (Trestbps)
     ggplot(df, aes(x = age, y = trtbps)) +
       geom_tile(aes(fill = ..density..),stat="density2d",alpha=1) +
       scale_fill_gradientn(colors = color_theme) +
       labs(title = 'Heatmap: Age vs. Resting Blood Pressure', x = 'Age', y = 'Resting Blood Pressure (Trestbps)')
```

2.5.2. Heatmap: Serum Cholesterol Level (Chol) vs Max Heart Rate (Thalach)

```
[ ]  color_theme <- colorRampPalette(c('darkorange', 'orange', 'lightcoral', 'lightsalmon', 'mistyrose'))(50)

     #Heatmap for Serum Cholesterol Level (Chol) and Max Heart Rate (Thalach)

     ggplot(df, aes(x = chol, y = thalachh)) +
       geom_tile(aes(fill = ..density..),stat="density2d",alpha=1) +
       scale_fill_gradientn(colors = color_theme) +
       labs(title = 'Serum Cholesterol Level (Chol) vs Max Heart Rate (Thalach)', x = 'Cholesterol', y = 'Max Heart Rate')
```

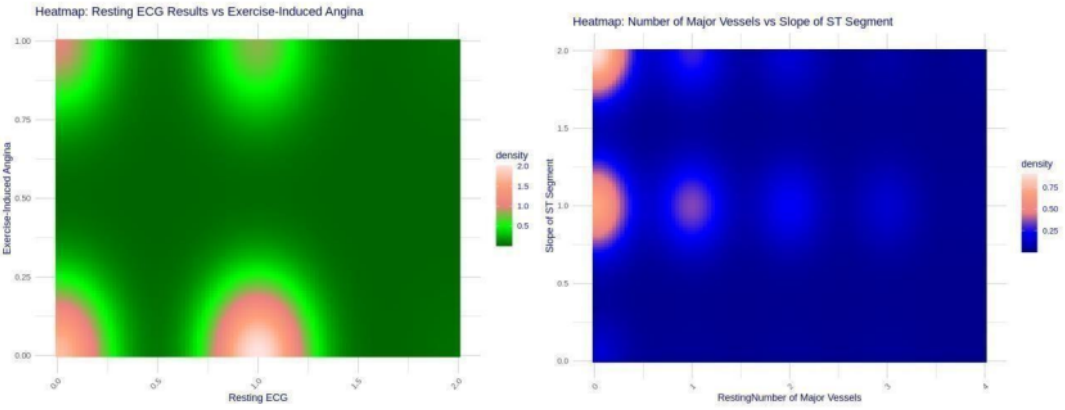


2.5.3. Heatmap: Resting ECG Results vs Exercise-Induced Angina

```
color_theme <- colorRampPalette(c('darkgreen', 'green', 'lightcoral', 'lightsalmon', 'mistyrose'))(50)

# Heatmap for Resting ECG Results (Restecg) and Exercise-Induced Angina (Exang)

ggplot(df, aes(x = restecg, y = exng)) +
  geom_tile(aes(fill = ..density..),stat="density2d",alpha=1) +
  scale_fill_gradientn(colors = color_theme) +
  labs(title = 'Heatmap: Resting ECG Results vs Exercise-Induced Angina', x = 'Resting ECG', y = 'Exercise-Induced Angina')
```

2.5.4. Heatmap: Number of Major Vessels vs Slope of ST Segment

```
color_theme <- colorRampPalette(c('darkblue', 'blue', 'lightcoral', 'lightsalmon', 'mistyrose'))(50)

# Heatmap for Number of Major Vessels (Ca) and Slope of ST Segment (Slope)


ggplot(df, aes(x = caa, y = slp)) +
  geom_tile(aes(fill = ..density..),stat="density2d",alpha=1) +
  scale_fill_gradientn(colors = color_theme) +
  labs(title = 'Heatmap: Number of Major Vessels vs Slope of ST Segment', x = 'RestingNumber of Major Vessels', y = 'Slope of ST Segment')
```

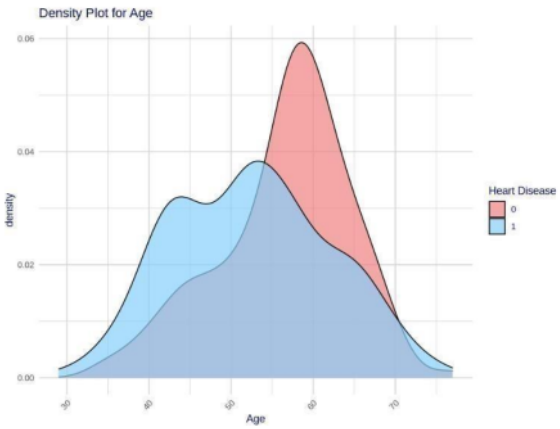


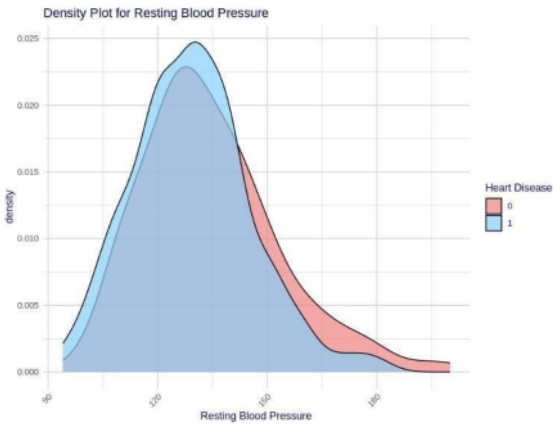
These are the heatmaps we can create based on the dataset.

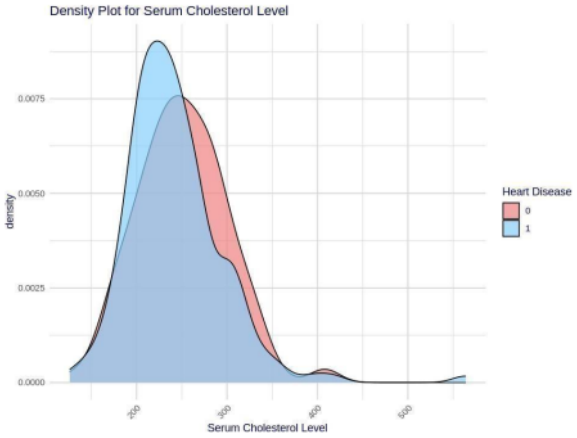
2.6. Density Plots

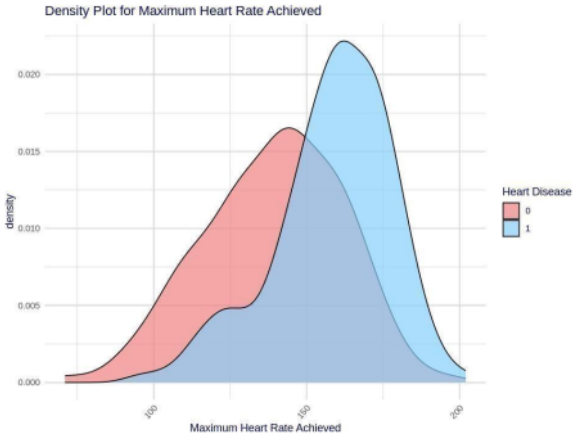
A density plot in data visualization is like a smooth line that shows the distribution of data values. It's like a fancier version of a histogram, giving you a sense of how data is spread out. Instead of using bars like a histogram, a density plot uses a curve to represent the overall shape of the data. The height of the curve at any point indicates the likelihood of finding a data point there. So, you can see where the data is concentrated or spread out. It's helpful for understanding patterns and trends in your data, especially when you want a more continuous and nuanced view of the distribution. These are the density plots we can create based on the dataset.

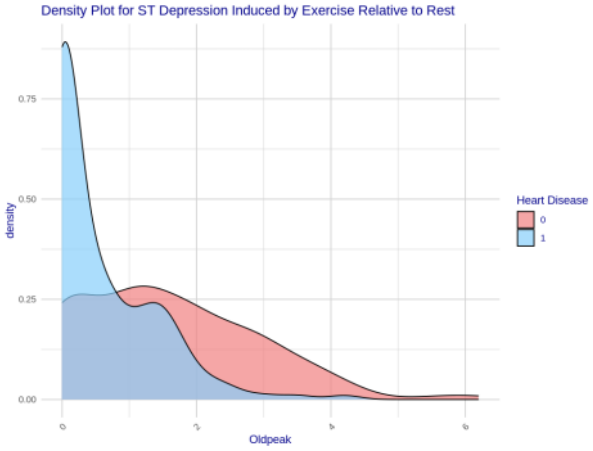
```
[ ] # Load necessary libraries
    library(ggplot2)
    library(dplyr)

    # Load the CSV file
    df <- read.csv("heart.csv")

    # Set a colorful theme
    theme_set(theme_minimal() +
                theme(text = element_text(color = 'darkblue'),
                      axis.text.x = element_text(angle = 45, vjust = 1, hjust = 1),
                      panel.grid.major = element_line(color = 'lightgray'),
                      panel.grid.minor = element_line(color = 'lightgray')))


    # List of variables to create density plots for
    variables <- c('age', 'trtbps', 'chol', 'thalachh', 'oldpeak', 'caa', 'thall', 'cp', 'slp', 'thall', 'output')

    # Set up a color palette
    color_palette <- c('0' = 'lightcoral', '1' = 'lightskyblue')

    # Create density plots using a for loop
    for (variable in variables) {
      a <- ggplot(df, aes(x = get(variable), fill = factor(output))) +
        geom_density(alpha = 0.7) +
        labs(title = paste('Density Plot for', variable), x = variable, fill = 'Heart Disease') +
        scale_fill_manual(values = color_palette)
      print(a)
    }
```

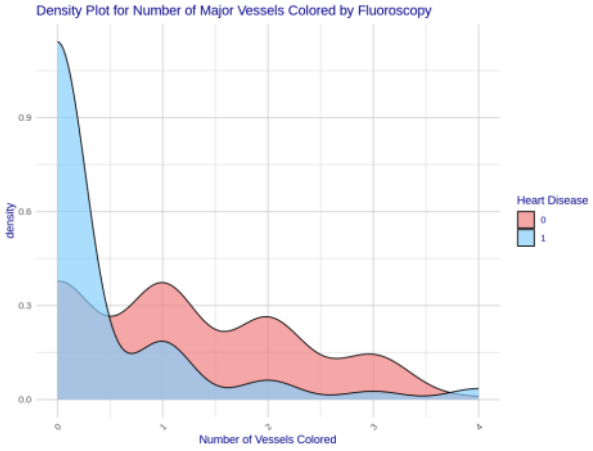

Density Plot for Age
density
Age
Heart Disease
0
1

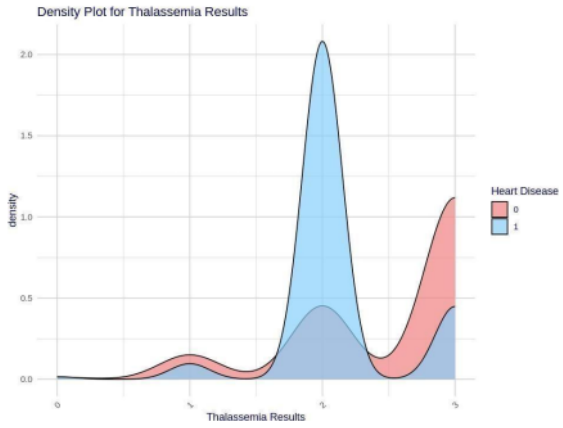

Density Plot for Resting Blood Pressure
density
Resting Blood Pressure
Heart Disease
0
1

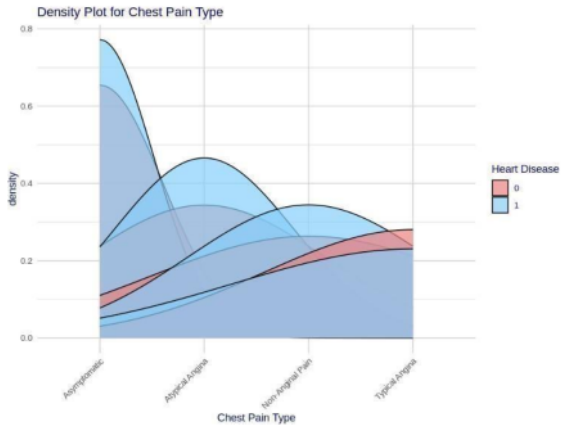

Density Plot for Serum Cholesterol Level
density
Serum Cholesterol Level
Heart Disease
0
1

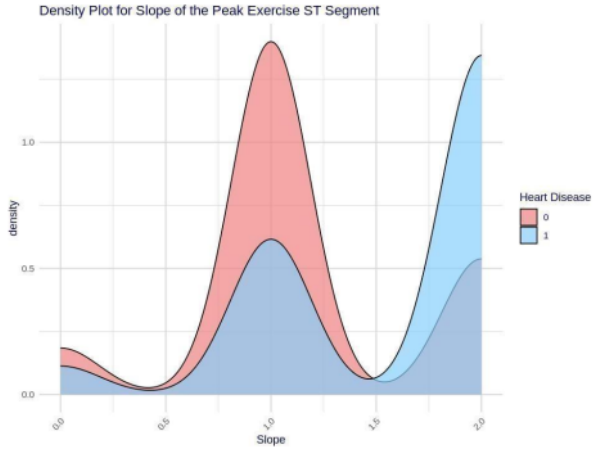

Density Plot for Maximum Heart Rate Achieved
density
Maximum Heart Rate Achieved
Heart Disease
0
1

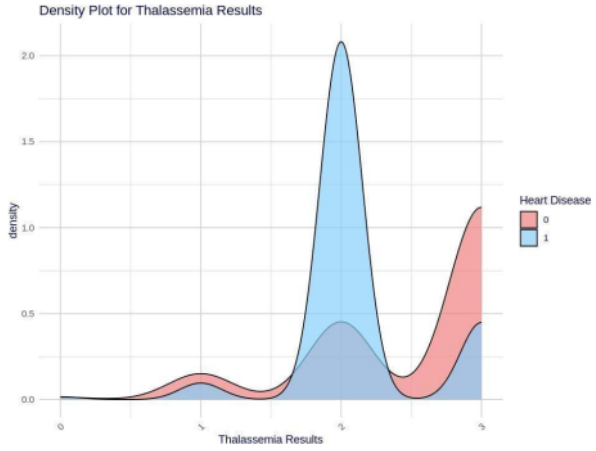

Density Plot for ST Depression Induced by Exercise Relative to Rest
density
Oldpeak
Heart Disease
0
1

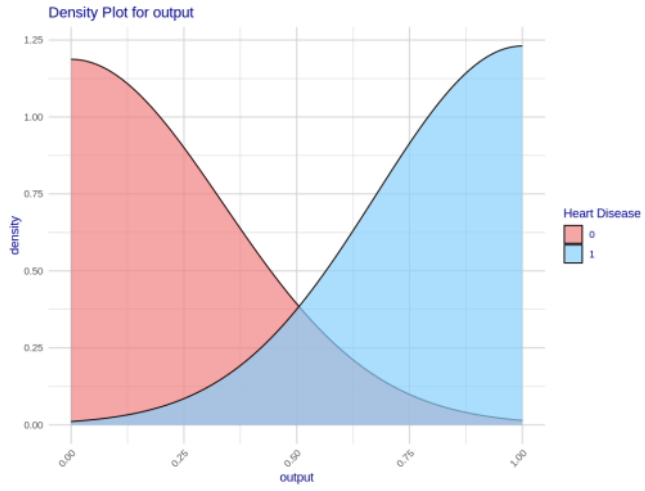

Density Plot for Number of Major Vessels Colored by Fluoroscopy
density
Number of Vessels Colored
Heart Disease
0
1


Density Plot for Thalassemia Results
density
Thalassemia Results
Heart Disease
0
1


Density Plot for Chest Pain Type
density
Asymptomatic
Atypical Angina
Non-Anginal Pain
Typical Angina
Chest Pain Type
Heart Disease
0
1


Density Plot for Slope of the Peak Exercise ST Segment
density
Slope
Heart Disease
0
1

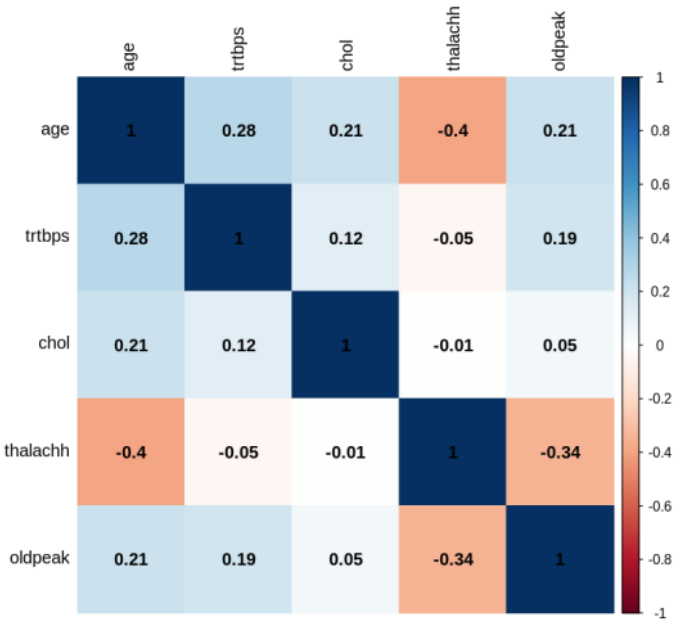

Density Plot for Thalassemia Results
density
Thalassemia Results
Heart Disease
0
1


Density Plot for output
density
output
Heart Disease
0
1

2.7. Correlation Matrices.

A correlation matrix serves as a tabular representation showcasing correlation coefficients among various variables. This matrix illustrates the correlation between every conceivable pair of values within a dataset. It proves to be a potent instrument for summarizing extensive datasets, facilitating the identification and visualization of patterns inherent in the provided data. Employing a correlation matrix aids in summarizing vast datasets, discerning patterns, and informing decision-making based on the observed correlations.

```
[ ]  # Install and load the corrplot library
     if (!requireNamespace("corrplot", quietly = TRUE)) {
       install.packages("corrplot")
     }
     library(corrplot)

     # Create a correlation matrix
     cor_matrix <- cor(df[, c("age", "trtbps", "chol", "thalachh", "oldpeak")])

     # Plot a correlation heatmap with numerical values inside the matrix
     corrplot(cor_matrix, method = "color", addCoef.col = "black", tl.col = "black")
```



3. Model Development

3.1. Pre-Processing

3.1.1. Checking for NULL/Missing Values.

The first step in data pre-processing is checking for NULL/missing values in the dataset. The code below checks for NULL values in the dataset. The output indicates that the dataset doesn't have any NULL/missing values in any of the attributes/columns.

```
# Checking for missing values
print(colSums(is.na(df)))
```

```
     age      sex       cp   trtbps     chol      fbs  restecg thalachh
       0        0        0        0        0        0        0        0
   exng  oldpeak      slp      caa    thall   output
       0        0        0        0        0        0
```

3.1.2. Checking for Duplicates

The second step in data pre-processing is checking for duplicates in the dataset. The code below verifies whether any duplicates exist in the dataset. According to the output, there is one duplicate value in the dataset.

```
[8]  # Checking for duplicated rows
     print(sum(duplicated(df)))
```

```
[1] 1
```

The code below identifies and displays duplicate rows in the dataset. According to the output, row 165 is a duplicate of row 166 in the dataset.

```
[9]  # Displaying duplicated rows
     print(df[duplicated(df), ])
```

```
     age  sex               cp trtbps chol   fbs  restecg thalachh exng oldpeak
165   38 Male Non-Anginal Pain    138  175 FALSE STT Wave      173   No       0
    slp caa        thall output
165   2   4 Fixed Defect      1
```

3.1.3. Removing Duplicates.

The code below is used to remove duplicate rows in the dataset.

```
[10] # Removing duplicated rows
     df <- df[!duplicated(df), ]
```

The code below is used to display the updated dataset.

```
[11] # Displaying the updated dataframe shape
     print(dim(df))

     [1] 302  14
```

Now the dataset only has 302 rows; previously, it had 303 rows.

3.1.4. Statistics of Cleaned Data

After cleaning the dataset, observe the dataset. The code below is used to display a summary of the dataset.

```
# Displaying summary statistics
print(summary(df))
```

```
      age            sex                      cp          trtbps
 Min.   :29.00   Female: 96   Asymptomatic    : 23   Min.   : 94.0
 1st Qu.:48.00   Male  :206   Atypical Angina : 50   1st Qu.:120.0
 Median :55.50                Non-Anginal Pain: 86   Median :130.0
 Mean   :54.42                Typical Angina  :143   Mean   :131.6
 3rd Qu.:61.00                                       3rd Qu.:140.0
 Max.   :77.00                                       Max.   :200.0
      chol          fbs             restecg       thalachh      exng
 Min.   :126.0   Mode :logical   LVH     :  4   Min.   : 71.0   No :203
 1st Qu.:211.0   FALSE:257       Normal  :147   1st Qu.:133.2   Yes: 99
 Median :240.5   TRUE :45        STT Wave:151   Median :152.5
 Mean   :246.5                                  Mean   :149.6
 3rd Qu.:274.8                                  3rd Qu.:166.0
 Max.   :564.0                                  Max.   :202.0
    oldpeak          slp             caa                    thall
 Min.   :0.000   Min.   :0.000   Min.   :0.0000   Fixed Defect     :165
 1st Qu.:0.000   1st Qu.:1.000   1st Qu.:0.0000   Normal           : 20
 Median :0.800   Median :1.000   Median :0.0000   Reversable Defect:117
 Mean   :1.043   Mean   :1.397   Mean   :0.7185
 3rd Qu.:1.600   3rd Qu.:2.000   3rd Qu.:1.0000
 Max.   :6.200   Max.   :2.000   Max.   :4.0000
    output
 Min.   :0.000
 1st Qu.:0.000
 Median :1.000
 Mean   :0.543
 3rd Qu.:1.000
 Max.   :1.000
```

```
[13] str(df)

'data.frame':   302 obs. of  14 variables:
 $ age     : int  63 37 41 56 57 57 56 44 52 57 ...
 $ sex     : Factor w/ 2 levels "Female","Male": 2 2 1 2 1 2 1 2 2 2 ...
 $ cp      : Factor w/ 4 levels "Asymptomatic",..: 1 3 2 2 4 4 2 2 3 3 ...
 $ trtbps  : int  145 130 130 120 120 140 140 120 172 150 ...
 $ chol    : int  233 250 204 236 354 192 294 263 199 168 ...
 $ fbs     : logi  TRUE FALSE FALSE FALSE FALSE FALSE ...
 $ restecg : Factor w/ 3 levels "LVH","Normal",..: 2 3 2 3 3 3 2 3 3 3 ...
 $ thalachh: int  150 187 172 178 163 148 153 173 162 174 ...
 $ exng    : Factor w/ 2 levels "No","Yes": 1 1 1 1 2 1 1 1 1 1 ...
 $ oldpeak : num  2.3 3.5 1.4 0.8 0.6 0.4 1.3 0 0.5 1.6 ...
 $ slp     : int  0 0 2 2 2 1 1 2 2 2 ...
 $ caa     : int  0 0 0 0 0 0 0 0 0 0 ...
 $ thall   : Factor w/ 3 levels "Fixed Defect",..: 2 1 1 1 1 2 1 3 3 1 ...
 $ output  : int  1 1 1 1 1 1 1 1 1 1 ...
```

The following code is used to display the head of the dataset.

```
[15] head(df)
```

A data.frame: 6 × 14

	age	sex	cp	trtbps	chol	fbs	restecg	thalachh	exng	oldpeak	slp	caa	thall	output
	<int>	<fct>	<fct>	<int>	<int>	<lgl>	<fct>	<int>	<fct>	<dbl>	<int>	<int>	<fct>	<int>
1	63	Male	Asymptomatic	145	233	TRUE	Normal	150	No	2.3	0	0	Normal	1
2	37	Male	Non-Anginal Pain	130	250	FALSE	STT Wave	187	No	3.5	0	0	Fixed Defect	1
3	41	Female	Atypical Angina	130	204	FALSE	Normal	172	No	1.4	2	0	Fixed Defect	1
4	56	Male	Atypical Angina	120	236	FALSE	STT Wave	178	No	0.8	2	0	Fixed Defect	1
5	57	Female	Typical Angina	120	354	FALSE	STT Wave	163	Yes	0.6	2	0	Fixed Defect	1
6	57	Male	Typical Angina	140	192	FALSE	STT Wave	148	No	0.4	1	0	Normal	1

The code below is used to display the tail of the dataset.

```
[14] tail(df)
```

A data.frame: 6 × 14

	age	sex	cp	trtbps	chol	fbs	restecg	thalachh	exng	oldpeak	slp	caa	thall	output
	<int>	<fct>	<fct>	<int>	<int>	<lgl>	<fct>	<int>	<fct>	<dbl>	<int>	<int>	<fct>	<int>
298	59	Male	Typical Angina	164	176	TRUE	Normal	90	No	1.0	1	2	Normal	0
299	57	Female	Typical Angina	140	241	FALSE	STT Wave	123	Yes	0.2	1	0	Reversable Defect	0
300	45	Male	Asymptomatic	110	264	FALSE	STT Wave	132	No	1.2	1	0	Reversable Defect	0
301	68	Male	Typical Angina	144	193	TRUE	STT Wave	141	No	3.4	1	2	Reversable Defect	0
302	57	Male	Typical Angina	130	131	FALSE	STT Wave	115	Yes	1.2	1	1	Reversable Defect	0
303	57	Female	Atypical Angina	130	236	FALSE	Normal	174	No	0.0	1	1	Fixed Defect	0

3.2. Dataset Splitting

To construct a dependable ML model, it is required to split the data into training, validation, and test subsets. Failure to do so may introduce bias in our outcomes, leading to a deceptive perception of

improved model accuracy. Consequently, we have partitioned the data into two sets, denoted as x_train and y_train for the train data, and x_test and y_test for the test data.

Split Ratio is **0.8**

```
[ ]

    set.seed(123)

    split <- sample.split(df$output, SplitRatio = 0.8)

    # Create training and testing sets
    train_data <- subset(df, split == TRUE)
    test_data <- subset(df, split == FALSE)

    # Separate input and output variables for training and testing sets
    x_train <- train_data[, c("age", "sex", "trtbps", "chol", "fbs", "restecg", "thalachh", "exng", "oldpeak", "slp", "caa", "thall")]
    y_train <- train_data[, "output"]

    x_test <- test_data[, c("age", "sex", "trtbps", "chol", "fbs", "restecg", "thalachh", "exng", "oldpeak", "slp", "caa", "thall")]
    y_test <- test_data[, "output"]
```

3.3. K-NN algorithm

K-NN emerges as a frequently utilized technique in ML. Its basis is rooted in the idea that data points exhibiting similarities frequently exhibit similar labels or values.

```
[ ]  # Install and load the necessary library
     install.packages("class")
     library(class)

     # Create a KNN classifier
     k <- 3  # You can adjust the value of 'k'
     knn_model <- knn(train = as.matrix(x_train), test = as.matrix(x_test), cl = y_train, k = k)

     # Print accuracy on the training set
     train_accuracy <- sum(knn_model == y_train) / length(y_train)
     cat("Accuracy of KNN on trained dataset: ", train_accuracy, "\n")

     # Use the trained model to predict on the test set
     predictions <- knn(train = as.matrix(x_train), test = as.matrix(x_test), cl = y_train, k = k)

     # Print accuracy on the test set
     test_accuracy <- sum(predictions == y_test) / length(y_test)
     cat("Accuracy of KNN on test dataset: ", test_accuracy, "\n")
```

i. The first step is installing the 'class' package and then loading it into the environment. The 'class' package provides functions for various classification algorithms, including KNN.

ii. In the next phase, a K-nearest neighbors (KNN) classifier is established with a designated 'k' value, set at 3 in this instance. The model undergoes training using the provided training set (x_train and y_train), after which it is employed to assess the test set (x_test). The resultant model is retained in the variable knn_model.

iii. In the third phase, the accuracy of KNN model on train data is determined & displayed. This involves evaluating the accuracy by comparing the model's predicted values (knn_model) to the true labels (y_train), performing the necessary calculations, and presenting the outcome.

iv. In the fourth step the trained KNN model is applied to the test set to make predictions. The predicted labels are stored in the variable predictions.

v. In the concluding phase, we compute and display the accuracy of KNN model on test data. Just like the process used to determine training set accuracy, this involves comparing the predicted values (predictions) with the real labels (y_test), computing the accuracy, and showcasing the outcome.

The output of the above code would be:

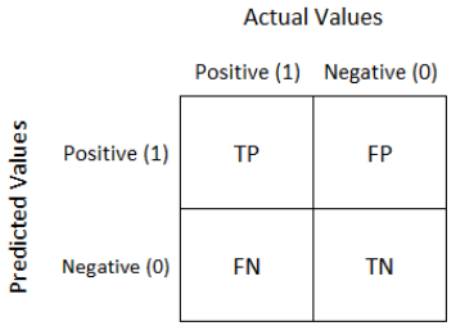
```
[ ]  # Print accuracy on the test set
     test_accuracy <- sum(predictions == y_test) / length(y_test)
     cat("Accuracy of KNN on test dataset: ", test_accuracy, "\n")
```

```
Installing package into '/usr/local/lib/R/site-library'
(as 'lib' is unspecified)

Warning message in `==.default`(knn_model, y_train):
"longer object length is not a multiple of shorter object length"
Warning message in is.na(e1) | is.na(e2):
"longer object length is not a multiple of shorter object length"
Accuracy of KNN on trained dataset:  0.5082645
Accuracy of KNN on test dataset:  0.5737705
```

We can see that the accuracy of KNN on trained dataset is 0.5082645 and accuracy of KNN on test dataset is 0.5737705.

3.3.1. Confusion matrix of k-NN

A confusion matrix serves as a tabular representation employed to assess the effectiveness of a classification algorithm. It presents a visual summary of how well a classification algorithm performs.




```r
# Install and load the necessary library
install.packages("class")
library(class)

# Create a KNN model
k <- 3  # You can adjust the value of 'k'
knn_model <- knn(train = as.matrix(x_train), test = as.matrix(x_test), cl = y_train, k = k)

# Combine levels of knn_model and y_test
combined_levels <- union(levels(factor(knn_model)), levels(factor(y_test)))

# Convert knn_model and y_test to factors with the same levels
knn_model <- factor(knn_model, levels = combined_levels)
y_test <- factor(y_test, levels = combined_levels)

# Create a confusion matrix
confusion_matrix <- confusionMatrix(knn_model, y_test)

# Print the confusion matrix
print(confusion_matrix)
```

i. The utilization of the "caret" package's confusionMatrix function is employed to generate a table summarizing the classification algorithm's performance. This function takes the predicted values (knn_model) and the actual values (y_test) as inputs to produce the confusion matrix.

ii. In conclusion, the console displays the confusion matrix, presenting details about the count of accurate positives, accurate negatives, erroneous positives, and erroneous negatives.

The output is:

```
Installing package into '/usr/local/lib/R/site-library'
(as 'lib' is unspecified)

Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0 13 11
         1 15 22

               Accuracy : 0.5738
                 95% CI : (0.4406, 0.6996)
    No Information Rate : 0.541
    P-Value [Acc > NIR] : 0.3513

                  Kappa : 0.1324

 Mcnemar's Test P-Value : 0.5563

            Sensitivity : 0.4643
            Specificity : 0.6667
         Pos Pred Value : 0.5417
         Neg Pred Value : 0.5946
             Prevalence : 0.4590
         Detection Rate : 0.2131
   Detection Prevalence : 0.3934
      Balanced Accuracy : 0.5655

       'Positive' Class : 0
```

From above the True Positive is 13, the False Positive is 11, theFalse Negative is 15 and True Negative is 22.

3.4. Linear Regression

```
[ ]  # Create a Linear Regression model
     linear_model <- lm(y_train ~ ., data = cbind(y_train, x_train))

     # Use the trained model to predict on the test set
     predictions_linear <- predict(linear_model, newdata = cbind(y_test, x_test))

     # Calculate the Mean Squared Error (MSE) on the test set
     mse_linear <- mean((predictions_linear - y_test)^2)
     cat("Mean Squared Error of Linear Regression on test dataset: ", mse_linear, "\n")
```

i. lm stands for linear model, and it's used to create a linear regression model.

ii. y_train ~ . specifies that the dependent variable y_train is modeled as a linear combination of all independent variables in x_train.

iii. cbind(y_train, x_train) combines the dependent and independent variables into a single

data frame.

iv. In second step ‘predict’ is used to make predictions using the trained linear regression model.

v. Incorporating the test set's dependent variable, y_test, and its independent variables, x_test, is achieved by merging them using the cbind function in the newdata set.

vi. Step three involves computing the Mean Squared Error (MSE) by determining avg. of squared variances among predicted values and real values.

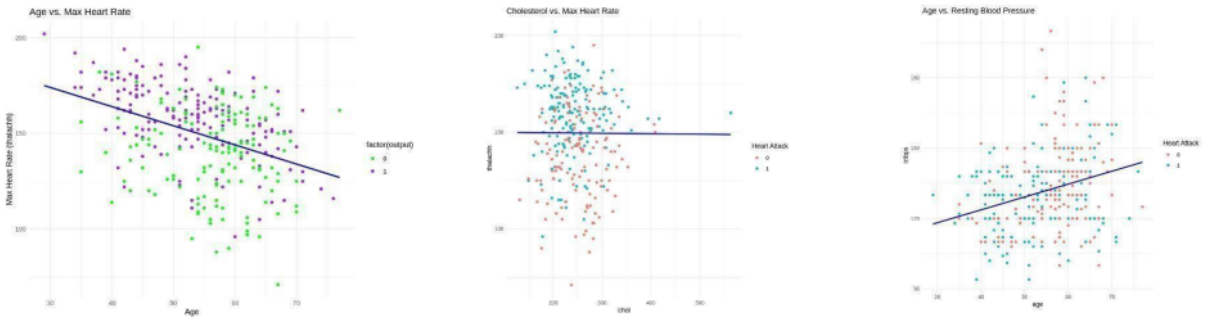
vii. The formula (predictions_linear - y_test)^2 computes the squared errors for each prediction.

viii. In the last stage, the script outputs the computed Mean Squared Error (MSE), offering a numerical assessment of the linear regression model's performance on the test set.

Output for code would be:

```
Mean Squared Error of Linear Regression on test dataset:  0.1578513
```

If we plot a scatter plot for this linear regression model than would be like this:



The technique of least squares is employed to establish the linear regression line, minimizing the total of squared vertical distances (residuals) between observed data points and the projected values along the y=mx+c slope. This optimal-fitting line encapsulates the overall pattern in the scatter plot, serving as a model for predictions or comprehending the relationship between the variables. Residuals, which are the vertical distances from data points to the line, quantify the accuracy of the model. The coefficient of determination gauges the goodness of fit, signifying the change in variance of dependent elucidated by independent variable. The linear

regression model, articulated as y=mx+c, furnishes a valuable tool for examining and foreseeing relationships between variables in astraightforward and understandable manner.

3.4. Logistic Regression

Logistic regression serves as a statistical technique utilized in data analysis, employing mathematical principles to reveal associations between two variables. It then utilizes these identified associations to predict the outcome of one variable based on the other. The resulting predictions are typically constrained to a finite set of possibilities, encompassing responses like affirmative or negative.

```
[ ] # Create a Logistic Regression model
    logistic_model <- glm(y_train ~ ., data = cbind(y_train, x_train), family = binomial)

    # Print accuracy on the training set
    train_accuracy_logistic <- sum((predict(logistic_model, type = "response", newdata = cbind(y_train, x_train)) > 0.5) == y_train) / length(y_train)
    cat("Accuracy of Logistic Regression on trained dataset: ", train_accuracy_logistic, "\n")

    # Use the trained model to predict on the test set
    predictions_logistic <- predict(logistic_model, type = "response", newdata = cbind(y_test, x_test))

    # Print accuracy on the test set
    test_accuracy_logistic <- sum((predictions_logistic > 0.5) == y_test) / length(y_test)
    cat("Accuracy of Logistic Regression on test dataset: ", test_accuracy_logistic, "\n")
```

i. The first line uses the glm function to create a logistic regression model (logistic_model).
ii. The formula y_train ~ . specifies that the response variable is y_train, and the predictors are all other variables in the dataset.
iii. data = cbind(y_train, x_train) combines the response variable y_train with the predictor variables x_train.
iv. family = binomial indicates that it's a logistic regression model, suitable for binary classification problems.
v. The second line calculates precision of logistic regression model on the train data.
vi. Predict function generates predicted probabilities (type = "response") for the trainingset based on the trained model.
vii. These probabilities are compared to a threshold of 0.5 to classify instances as 0 or 1.
viii. The accuracy is calculated by comparing the predicted classifications to the actual values (y_train).
ix. The predict function is used again to generate predicted probabilities for the test set based on the trained logistic regression model.

x. Similar to the training set, predicted probabilities are compared to a threshold of 0.5 to classify instances as 0 or 1.

xi. The accuracy on the test set is calculated by comparing the predicted classifications to the actual values (y_test).

xii. The last lines print the accuracy results on both train and test data. The analysis provides information about how well logistic regression model performs on both datasets.

The output for this code would be:

The accuracy of Logistic Regression on trained dataset is 0.8347107 and accuracy of Logistic Regression on test dataset is 0.7540984.

```
# Print accuracy on the test set
test_accuracy_logistic <- sum((predictions_logistic > 0.5) == y_test) / length(y_test)
cat("Accuracy of Logistic Regression on test dataset: ", test_accuracy_logistic, "\n")
```

```
Accuracy of Logistic Regression on trained dataset:  0.8347107
Accuracy of Logistic Regression on test dataset:  0.7540984
```

3.5. NB Classifier

NB represents straightforward learning algorithm that applies Bayes' rule whilemaking a robust assumption: that the attributes are conditionally independent when considering the class.

```
[ ] # Create a Naive Bayes model
    nb_model <- naiveBayes(y_train ~ ., data = cbind(y_train, x_train))

    # Print accuracy on the training set
    train_accuracy_nb <- sum(predict(nb_model, newdata = cbind(y_train, x_train)) == y_train) / length(y_train)
    cat("Accuracy of Naive Bayes on trained dataset: ", train_accuracy_nb, "\n")

    # Use the trained model to predict on the test set
    predictions_nb <- predict(nb_model, newdata = cbind(y_test, x_test))

    # Print accuracy on the test set
    test_accuracy_nb <- sum(predictions_nb == y_test) / length(y_test)
    cat("Accuracy of Naive Bayes on test dataset: ", test_accuracy_nb, "\n")
```

i. ‘naiveBayes’: It is used to create a Naive Bayes classifier.

ii. ‘y_train ~ .’: This formula specifies that the target variable y_train is to be predicted based on all other variables represented by the . (dot), which typically includes the features in the dataset.

iii. data = cbind(y_train, x_train): It combines the target variable y_train and the feature matrix x_train into a single data frame for model training.

iv. predict: This function is used to make predictions on the training set using the trained Naive Bayes model.

v. sum(...) == y_train: It evaluates the anticipated outcomes by contrasting them with the genuine target values (y_train) and aggregates the accurate predictions.

vi. length(y_train): It calculates the total no. of train occurrences.

vii. Result is exactness on training set, which is then printed to the console.

viii. predict: This functionality is employed to generate forecasts for the test dataset utilizing the Naive Bayes model that has been trained beforehand.

ix. sum(predictions_nb == y_test): It calculates the number of correct predictions on the test set.

x. length(y_test): It calculates total no. of occurrences in test set.

```
# Print accuracy on the test set
test_accuracy_nb <- sum(predictions_nb == y_test) / length(y_test)
cat("Accuracy of Naive Bayes on test dataset: ", test_accuracy_nb, "\n")
```

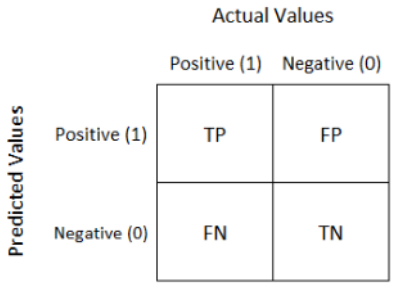
```
Accuracy of Naive Bayes on trained dataset:  0.8305785
Accuracy of Naive Bayes on test dataset:  0.7540984
```

Accuracy of Naive Bayes on trained dataset is 0.8305785 and on testdataset is 0.7540984.

3.5.1. Confusion Matrix for Naive Bayes Classifier

i. A confusion matrix serves as a tabular representation employed for assessing the efficacy of a classification algorithm. It visually presents and condenses the outcomes of a classification algorithm's performance.

ii. The figure presented below depicts a generic confusion matrix.



```r
# Create a Naive Bayes model
nb_model <- naiveBayes(y_train ~ ., data = cbind(y_train, x_train))

# Use the trained model to predict on the test set
predictions_nb <- predict(nb_model, newdata = cbind(y_test, x_test))

# Combine levels of predictions_nb and y_test
combined_levels <- union(levels(predictions_nb), levels(y_test))

# Convert predictions_nb and y_test to factors with the same levels
predictions_nb <- factor(predictions_nb, levels = combined_levels)
y_test <- factor(y_test, levels = combined_levels)

# Create a confusion matrix
confusion_matrix <- confusionMatrix(predictions_nb, y_test)

# Print the confusion matrix
print(confusion_matrix)
```

i. The utilization of the confusion matrix function within the caret package facilitates the generation of a summary reflecting the performance of the classification model. This summary involves a comparison between the predicted and actual class labels.

ii. Ultimately, the console outputs the confusion matrix, providing a breakdown of counts for various predictions.

The output would be:

```
Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0 20  7
         1  8 26

               Accuracy : 0.7541
                 95% CI : (0.6271, 0.8554)
    No Information Rate : 0.541
    P-Value [Acc > NIR] : 0.0004963

                  Kappa : 0.5035

 Mcnemar's Test P-Value : 1.0000000

            Sensitivity : 0.7143
            Specificity : 0.7879
         Pos Pred Value : 0.7407
         Neg Pred Value : 0.7647
             Prevalence : 0.4590
         Detection Rate : 0.3279
   Detection Prevalence : 0.4426
      Balanced Accuracy : 0.7511

       'Positive' Class : 0
```

According to the confusion matrix the True Positive is 20, the False Positive is 7, the False Negative is 8 and True Negative is 26.

4. Evaluation

4.1. Algorithms Result Table

SL.NO.	ML Algorithms	Train	Test
1.	Linear Regression	0.1578513 (Mean Square Error)	
2.	Logistic Regression	0.8347107	0.7540984
3.	kNN Classifier	0.5082645	0.5737705
4.	Naive Bayes model	0.8305785	0.7540984

In terms of accuracy, the Logistic Regression model exhibits slightly higher performance on the training set (0.8347107 compared to 0.8305785 for Naive Bayes). However, both models achieve the same accuracy on the test set (0.7540984). Based on the provided accuracy scores,LR model performs slightly better on train data. It is crucial to highlight that relying solely on accuracy might not offer a complete summary of the model's efficiency.

The LR model only outperforms the NB model on the training set when the SplitRatio is set to 0.8. Changing the SplitRatio to 0.7 or 0.6 results in decreased accuracy. In summary, if accuracy is the sole criterion for comparison, the Logistic Regression model appears to have a slight edge on the training set when the SplitRatio is 0.8; however, altering the SplitRatio may favor the Naive Bayes model in achieving the highest accuracy.

4.2. Future Works

Future advancements in heart attack prediction and analysis could focus on improving predictive model accuracy and interpretability. Advanced machine learning algorithms, such as ensemble methodologies or architectures based on deep learning, could be instrumental in discerning intricate patterns within cardiovascular data. Furthermore, incorporating additional elements such as genetic markers, improved imaging data, or real-time physiological monitoring via wearable devices could provide a more thorough assessment of a person's cardiovascular health. Collaboration with healthcare experts and institutions would be required to collect diverse and large-scale datasets for robustmodel training and validation.

In addition, future research could focus on the application of these predictive models in real-world clinical settings. This entails conducting prospective studies to evaluate the models' efficacy in guiding early treatments and preventive measures. Integrating the models into electronic health record (EHR) systems and developing user-friendly interfaces for healthcare professionals will help to ensure widespread use. Ongoing research and development should also prioritize addressing ethical concerns, including patient privacy, data security, and ensuring that predictive models are used in accordance with established medical recommendations. Continuous model performance monitoring and regular upgrades based on changing medical knowledge would be critical for keeping heart attack prediction and analysis systems relevant and successful in the dynamic healthcare situation.

4.3. Summary

In the pursuit of identifying the most effective algorithm for forecasting an individual's susceptibility to a heart attack, a crucial initial step involves comprehensive data preprocessing. This intricate procedure encompasses not just data cleaning but also involves data integration, transformation, and selection. The goal is to ensure the clarity and relevance of the data for the machine to derive precise insights.

The task of predicting heart attacks inherently falls under the domain of supervised learning, given that the outcome is predetermined. Within the spectrum of supervised algorithms, a diverse array is at our disposal, including KNN, Naive Bayes, logistic regression, and linear regression. The pivotal decision lies in selecting the algorithm that achieves the highest accuracy.

Upon meticulous examination, the Naive Bayes algorithm emerges as the frontrunner in our case, exhibiting superior accuracy compared to its counterparts (Logistic Regression only provides the highest accuracy when the Split Ratio is 0.8). Consequently, the selection of the Naive Bayes algorithm becomes crucial for post-processing unknown predictions, thereby enhancing the overall predictive capabilities in determining whether an individual is prone to a heart attack or not. This comprehensive analysis underscores the significance of informed algorithm selection andthe careful preprocessing of data in the pursuit of accurate predictions.

C00528114_Semester-Project

ORIGINALITY REPORT

0%	0%	0%	0%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

Exclude quotes	Off	Exclude matches	Off
Exclude bibliography	Off		