HEART DISEASE ANALYSIS

BHANU KOLLAPU
2ND YEAR DSSTCS
RAJAHMUNDRY

OBJECTIVES:

The objective of the analysis is to explore a dataset related to heart diseases, perform data preprocessing, detect outliers, transform data, and visualize relationships between various health parameters and the presence of heart disease. The aim is to derive insights and understand potential factors contributing to heart disease.

INTRODUCTION ABOUT THE DATA SET:

We're studying a dataset called heart-disease-uci from Kaggle. It mainly uses the Cleveland database and focuses on whether patients have heart disease. The "target" column indicates the presence of heart disease (0 for no, 1 for yes). We're specifically looking at gender differences among patients with heart disease. Here are some important terms:

Age: Person's age in years

Sex: Person's gender (1 for male, 0 for female)

Chest Pain (cp): Different types of chest pain experienced

Resting Blood Pressure (trestbps): Blood pressure when admitted to the hospital

Cholesterol (chol): Cholesterol level in mg/dl

Fasting Blood Sugar (fbs): Blood sugar level (1 if > 120 mg/dl, 0 otherwise)

Resting Electrocardiographic Measurement (restecg): Electrical activity of the heart at rest

Maximum Heart Rate Achieved (thalach): Person's highest heart rate during a test

Exercise-Induced Angina (exang): Whether angina was induced by exercise

ST Depression Induced by Exercise (oldpeak): Changes in the ECG due to exercise

Slope of Peak Exercise ST Segment (slope): Pattern of the peak exercise ST segment

Number of Major Vessels (ca): Count of major blood vessels

Thalassemia (thal): A blood disorder with different types

Target: Presence of heart disease (0 for no, 1 for yes)

HEART DISEASE ANALYSIS:

❖ STEP1: LOADING THE DATA

➢ **Code:**

Assume the dataset is named "data"

```
data<-read.csv(file.choose(),header = TRUE)
data
```

▪ The read.csv() function is used to load a CSV file. file.choose() opens a file dialog for the user to select the file interactively. The resulting data is stored in a variable called data

- Libraries
  - ✓ library(ggplot2)
  - ✓ library(tidyr)
  - ✓ library(dplyr)

❖ STEP2: EXPLORING THE DATA

- head(data): Displays the first few rows of the dataset.

nrow(data): Shows the number of rows in the dataset.

ncol(data): Indicates the number of columns in the dataset.

str(data): Provides a concise summary of the structure of the dataset, including data types.

summary(data): Generates summary statistics such as mean, median, min, max, etc., for each variable in the dataset.

names(data): Displays the column names of the dataset.

❖ STEP 3: HANDLE NULL VALUES

- Code:

```
colSums(is.na(data))
```

  - This code provides a numeric vector where each element represents the count of missing values in the respective column of the dataset.

- Code:

```
most_frequent<-names(which.max(table(data$thal)))
data<-data%>% mutate(thal=
ifelse(thal==most_frequent,NA,thal))
most_frequent<-names(which.max(table(data$thal)))
data <- data %>% mutate(thal =
ifelse(is.na(thal), most_frequent,thal))
data

most_frequent<-names(which.max(table(data$slope)))
data<-data%>% mutate(slope=
ifelse(slope==most_frequent,NA,slope))
most_frequent<-names(which.max(table(data$slope)))
data <- data %>% mutate(slope =
ifelse(is.na(slope), most_frequent,slope))
```

```
most_frequent<-names(which.max(table(data$fbs)))
data <- data %>% mutate(fbs =
ifelse(is.na(fbs), most_frequent, fbs))

most_frequent<-names(which.max(table(data$exang)))
data <- data %>% mutate(exang=
ifelse(is.na(exang), most_frequent,exang))

most_frequent<-names(which.max(table(data$ca)))
data <- data %>% mutate(ca =
 ifelse(is.na(ca), most_frequent, ca))
```

- The code identifies the most frequent value in specific columns (thal,exang,ca,fbs,slope) and temporarily replaces those values with NA.
- It recalculates the most frequent value after treating the temporary NAs.
- It finally fills the original NAs in those columns with the recalculated most frequent value, except for 'oldpeak', which uses the mean value to replace missing entries.

  ```
  mean_value <- mean(data$oldpeak, na.rm = TRUE)
  data$oldpeak[is.na(data$oldpeak)]<-mean_value
  ```

- Here the code identifies the duplicates and replaces them with NA's and recalculates the and fills them in the place of temporary NA's

❖ STEP 4: DUPLICATES

➢ <u>Code</u>:

```
sum(duplicated(data))
```

- This code identifies the duplicates in the dataset
- There are no duplicates present in the dataset

❖ STEP 5: REMOVING COLUMNS

➢ Code:

```
data<-subset(data,select = -dataset)
data
```

- specifies that the 'dataset' column should be excluded from the subset. The minus sign (-) before 'dataset' indicates that this column should not be included in the resulting subset.

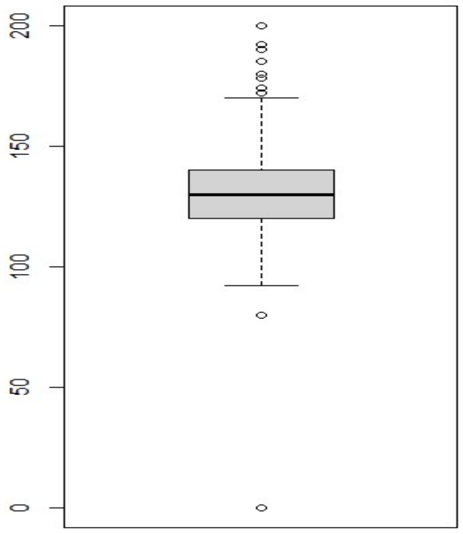
❖ STEP 5: OUTLIERS

➢ Code:

```
Library(ggplot2)
boxplot(data$age), boxplot(data$trestbps),
boxplot(data$chol), boxplot(data$thalch),
boxplot(data$exang), boxplot(data$oldpeak),
boxplot(data$target)

boxplot(data$trestbps)
```

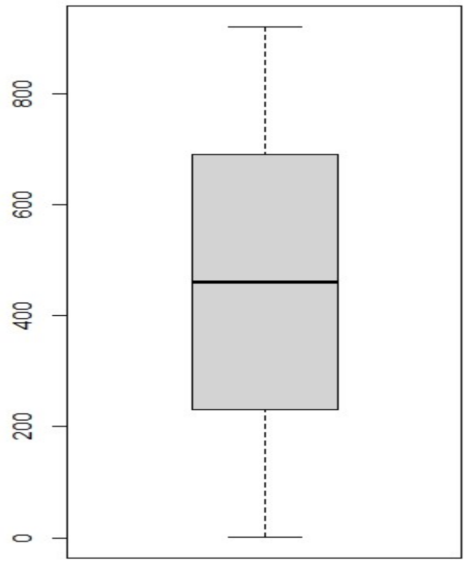
- Boxplots display the distribution of values in each column, showing median, quartiles, outliers, etc.



Outlier Detection using Z-Score

```
z_score<-scale(data$trestbps)
outliers<-which(abs(z_score)>3)
outliers
removeoutliers<-which(abs(z_score)<3)
removeoutliers
boxplot(removeoutliers)
```

- Z-score calculation and subsequent identification of outliers are performed for specific columns to analyze and visualize potential outliers through boxplots without those extreme values.



❖ STEP 6: TRANSFORMATION

➢ Code:

```
data$sex[data$sex == 'Male']<- 1
data$sex[data$sex=='Female']<-0
View(data)
```

```
data$fbs[data$fbs == 'TRUE']<- 1
data$fbs[data$fbs=='FALSE']<-0
View(data)

data$exang[data$exang == 'TRUE']<- 1
data$exang[data$exang=='FALSE']<-0
View(data)
```

- Here, the code replaces the occurances male to 1 and female to 0 in sex column and replaces true with 1 and false with 0 in fbs column and exang column

❖ STEP 7: STANDARDIZATION

- Standardization rescales the values of a numeric column so that they have a mean of 0 and a standard deviation of 1 (assuming a normal distribution). This process helps to compare different variables on the same scale
- Calculating the mean and standard deviation helps in understanding the distribution of the 'age' column. Mean represents the average value, while standard deviation measures the spread of values around the mean.

➢ Code:

```
# Calculate mean and standard deviation
mean_value <- mean(data$age)
sd_value <- sd(data$age)

# Standardize the column
data$standardized_age <- (data$age - mean_value) / sd_value
data$standardized_age
```

- The resulting standardized values are stored in a new column named 'standardized_age' in the data dataset, containing the standardized 'age' values that can be utilized for analysis or modeling purposes.

❖ STEP 8: NORMALIZATION

- Normalization is a scaling technique that transforms data to a common scale, often between 0 and 1. It maintains the relative relationships between values while scaling them down to a standardized range.

➢ Code:

```
# Calculate minimum and maximum values
min_value <- min(data$age)
max_value <- max(data$age)
```

- This code finds the minimum and maximum values in the ‘age’ column

```
# Normalize the column
data$normalized_age<- (data$age) / (max_value - min_value)
data$normalized_age
```

- Displays the resulting normalized values in a new column named 'normalized_age' in the data dataset.
- The resulting normalized values are stored in a new column named 'normalized_age' within the data dataset. This column contains the normalized 'age' values, helpful for certain algorithms or analyses that require inputs to be within a specific range, like [0, 1].

❖ STEP 9: ADDING COLUMN TO DATASET

➢ Code:

```
data$oldpeople <-data$age> 60
View(data)
```

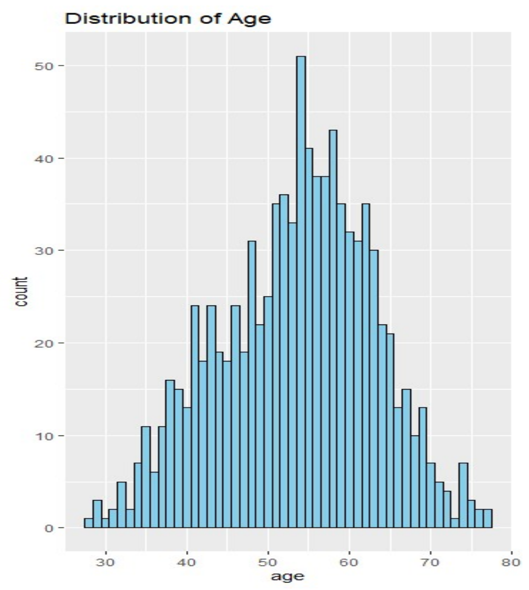
- This code introduces a new column named 'oldpeople' to the dataset data based on a condition related to the 'age' column.
- The 'oldpeople' column now contains boolean values (TRUE or FALSE) indicating whether each individual in the dataset is classified as 'old' (age greater than 60) or not

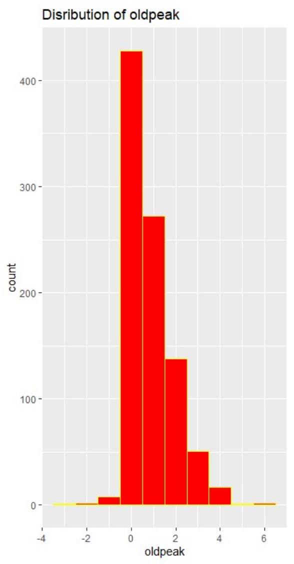
❖ STEP 10: DATA VISUALIZATION

- Exploring distributions, relationships, and patterns within the dataset, providing insights into potential associations between variables and their distributions.
- Histograms, scatterplots,boxplot, barplot, scatterplot matrix are used for data visualization.

➢ <u>Code:</u>

```
library(ggplot2)
# Visualizing distributions and relationships
# Example: Histogram for age
ggplot(data, aes(x = age)) +
  geom_histogram(binwidth = 1, fill = "skyblue", color =
"black") +
  labs(title = "Distribution of Age")
# Histogram for oldpeak
ggplot(data, aes(x = oldpeak))+
  geom_histogram(binwidth = 1, fill = "red", color = "yellow")
+
  labs(title = "Disribution of oldpeak ")
```



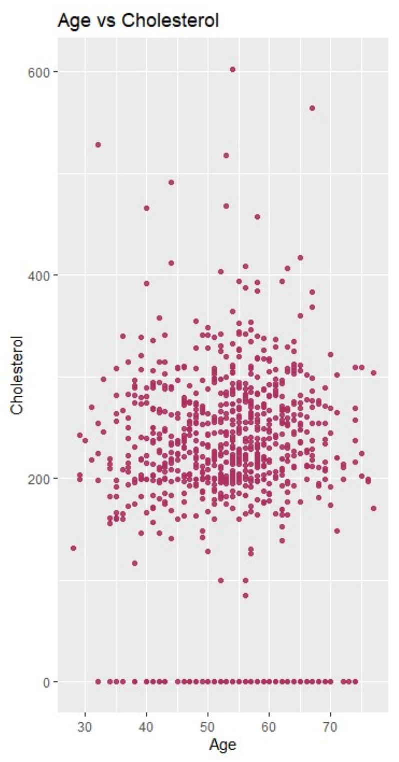


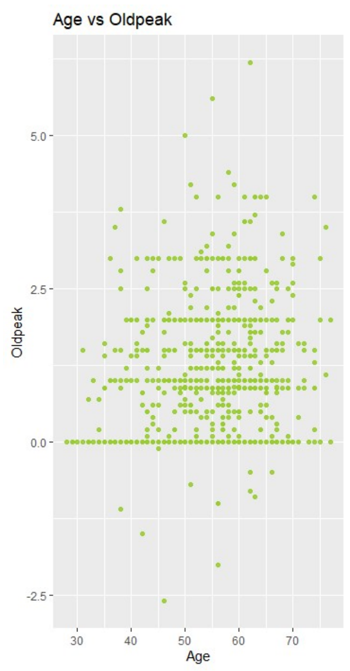
- This histogram visualization helps in understanding the distribution of ages and oldpeak within the dataset, showing the frequency or count of different age groups and oldpeaks present in the data.

➢ <u>Code</u>:

✓ The scatterplot matrix provides a matrix of scatterplots for pairwise relationships between selected columns, allowing visualization of correlations or patterns.

```
# Scatterplot of age vs cholesterol
ggplot(data, aes(x = age, y = chol)) +
  geom_point(color = "maroon", alpha = 0.9) +
  labs(title = "Age vs Cholesterol", x = "Age", y = "Cholesterol")
# Scatterplot of age vs oldpeak
ggplot(data, aes(x = age,, y = oldpeak)) +
  geom_point(color = "yellowgreen", alpha = 0.9) +
  labs(title = "Age vs Oldpeak", x ="Age", y = "Oldpeak")
```

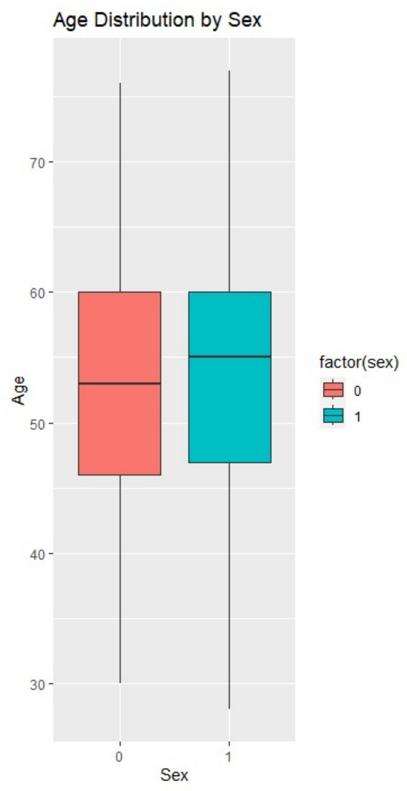




- Scatterplots (geom_point) depict relationships between variables. The code creates scatterplots for 'age' vs 'chol' and 'age' vs 'oldpeak'.

➢ <u>Code</u>:

```
# Boxplot of age by sex
ggplot(data, aes(x = sex, y = age, fill = factor(sex))) +
  geom_boxplot() +
  labs(title = "Age Distribution by Sex", x = "Sex", y = "Age")
```

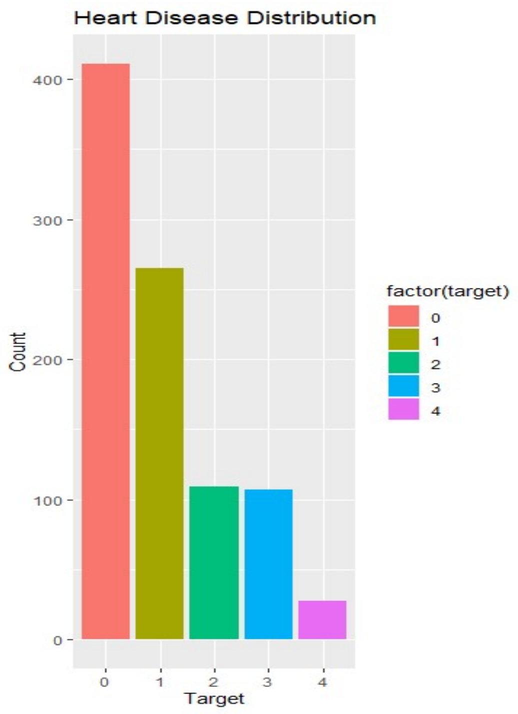


- The boxplot displays age distributions by sex.
- A barplot (geom_bar) visualizes the distribution of the 'target' variable, indicating counts for each category within the 'target' variable.

➢ Code:

Visualizing heart disease target variable

```
ggplot(data, aes(x = factor(target), fill = factor(target))) +
  geom_bar() +
  labs(title = "Heart Disease Distribution", x = "Target", y =
"Count")
```



- This code generates a bar plot representing the distribution of the 'target' variable related to heart disease in the dataset.
- The x-axis displays the categories of the 'target' variable.