Faculty of Computing

Group NO 06

Heart Disease Dataset Analysis

Table of Contents

1. Introduction

Cardiovascular disease is the leading cause of death worldwide, and early diagnosis and accurate risk identification are critical for effective treatment and prevention. "Heart Disease" is a very good dataset to identify heart disease risks based on patient characteristics. This dataset includes various types of medical data such as age, gender, type of chest pain, resting blood pressure, cholesterol level, fasting blood sugar, resting ECG results, peak heart rate, exercise-induced angina, old peak, slop, fluoroscopy-stained vessels. The parameters include thalassemia, and the target variable indicating the presence or absence of heart disease and the objective of this analysis is to effectively pre-process the data, gain significant insights and visualize and develop robust classification models for heart disease prediction. Additionally, clustering techniques will be used to uncover hidden patterns in the data set. Using these methods, we aim to improve our understanding of risk factors associated with heart disease and improve the accuracy of predictions, as well as ultimately contribute to better health outcomes for patients.

3. Using Library

Library	Description
readr	readr is used to read and write data in R. It provides faster and more user-friendly functions for data import, offering support for CSV, TSV, and other formats through functions like read_csv() and write_csv().
ggplot2	ggplot2 is a powerful visualization package based on the grammar of graphics. It allows for the creation of complex and aesthetic graphics by layering components such as points, lines, and text.
rpart.plot	rpart.plot is an extension of the rpart package used for recursive partitioning and regression trees. It provides functions to visualize decision trees created by rpart, making them easier to interpret.
caret	The caret package (short for Classification and Regression Training) is a comprehensive framework for building predictive models in R. It offers a unified interface to various machine learning algorithms, streamlining the process of model training, tuning, and evaluation.
pROC	pROC is designed for visualizing and analyzing receiver operating characteristic (ROC) curves. It is used for binary classification tasks to evaluate model performance and determine optimal classification thresholds.
tidyverse	tidyverse is a collection of R packages designed for data science, including ggplot2, dplyr, tidyr, and others that share a common design philosophy. It allows for seamless integration of data manipulation, visualization, and analysis functions.
cluster	The cluster package provides functions for cluster analysis, which involves grouping similar data points together. It includes various clustering algorithms and tools to visualize clustering results, useful for exploratory data analysis and pattern recognition.
factoextra	factoextra facilitates the visualization and interpretation of multivariate data analysis results. It is used to visualize clustering results, principal component analysis (PCA), and other multivariate techniques, enhancing the understanding of complex data sets.

3. Data Preprocessing/Cleaning

Cleaning for data preprocessing is an essential step in preparing the data set for analysis and modeling.

3.1 Handling Missing Data

First, it is important to identify missing values in the data set. Depending on the amount of missing data, different strategies can be applied. If the number of missing values by deletion is minimal, the affected rows or columns can be discarded.

For more significant gaps imputation can fill in missing values using statistical methods such as the mean, median, or mode of the relevant feature, or through more advanced techniques such as K-nearest neighbors (KNN) imputation.

```
# Identify missing values
missing <- colSums(is.na(data))
print(missing)

# Remove rows with missing values
data_clean <- na.omit(data)
print(data_clean)
```

```
> # Identify missing values
> missing <- colSums(is.na(data))
> print(missing)
                       age                        sex             chest_pain_type
                         0                          0                           0
      resting_blood_pressure                cholestoral          fasting_blood_sugar
                         0                          0                           0
                   rest_ecg             Max_heart_rate       exercise_induced_angina
                         0                          0                           0
                    oldpeak                      slope vessels_colored_by_flourosopy
                         0                          0                           0
                thalassemia                     target
                         0                          0
> # Remove rows with missing values
> data_clean <- na.omit(data)
> print(data_clean)
   age    sex  chest_pain_type resting_blood_pressure cholestoral    fasting_blood_sugar
1   52   Male   Typical angina                   125         212   Lower than 120 mg/ml
2   53   Male   Typical angina                   140         203 Greater than 120 mg/ml
3   70   Male   Typical angina                   145         174   Lower than 120 mg/ml
4   61   Male   Typical angina                   148         203   Lower than 120 mg/ml
5   62 Female   Typical angina                   138         294 Greater than 120 mg/ml
6   58 Female   Typical angina                   100         248   Lower than 120 mg/ml
7   58   Male   Typical angina                   114         318   Lower than 120 mg/ml
8   55   Male   Typical angina                   160         289   Lower than 120 mg/ml
```

3.2 Converting Categorical Variables

To ensure that our models correctly handle categorical data, we convert relevant columns (gender, type of chest pain, and others) into factors. This is important for correct model interpretation and accuracy.

```
# Preprocessing the data
# Convert categorical variables to factors
data$sex <- as.factor(data$sex)
data$chest_pain_type <- as.factor(data$chest_pain_type)
data$fasting_blood_sugar <- as.factor(data$fasting_blood_sugar)
data$rest_ecg <- as.factor(data$rest_ecg)
data$exercise_induced_angina <- as.factor(data$exercise_induced_angina)
data$slope <- as.factor(data$slope)
data$vessels_colored_by_flourosopy <- as.factor(data$vessels_colored_by_flourosopy)
data$thalassemia <- as.factor(data$thalassemia)
data$target <- as.factor(data$target)
```

```
> # Preprocessing the data
> # Convert categorical variables to factors
> data$sex <- as.factor(data$sex)
> data$chest_pain_type <- as.factor(data$chest_pain_type)
> data$fasting_blood_sugar <- as.factor(data$fasting_blood_sugar)
> data$rest_ecg <- as.factor(data$rest_ecg)
> data$exercise_induced_angina <- as.factor(data$exercise_induced_angina)
> data$slope <- as.factor(data$slope)
> data$vessels_colored_by_flourosopy <- as.factor(data$vessels_colored_by_flourosopy)
> data$thalassemia <- as.factor(data$thalassemia)
> data$target <- as.factor(data$target)
```

3.3 Splitting the Data

We split the dataset into training and testing sets, 80% of the data is used for training and 20% is reserved for testing. This partitioning allows us to evaluate the performance of our models on unseen data as well as their generalization.

```
# Split the dataset into training
set.seed(123)
train_index <- createDataPartition(data$target, p = 0.8, list = FALSE)
train_data <- data[train_index, ]
test_data <- data[-train_index, ]

# Display the dimensions of the training and testing sets
dim(train_data)
dim(test_data)
```

```
> set.seed(123)
> train_index <- createDataPartition(data$target, p = 0.8, list = FALSE)
> train_data <- data[train_index, ]
> test_data <- data[-train_index, ]
> # Display the dimensions of the training and testing sets
> dim(train_data)
[1] 821  14
> dim(test_data)
[1] 204  14
```

4. Important Insights and Visualizations

4.1 Visualizations of Key Attributes

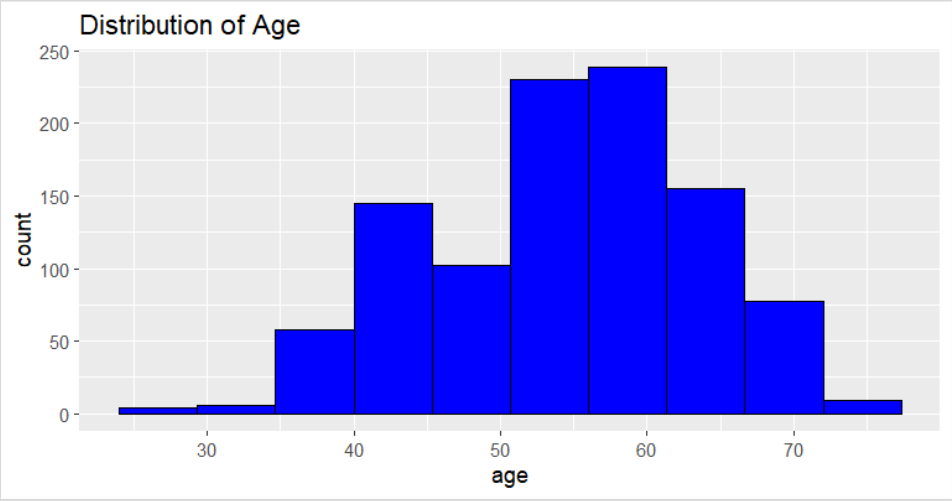
Visualizing the key properties of the data set, as well as helping to understand the distribution, relationships and potential patterns within the data.

Age Distribution

Histograms can be used to visualize the age distribution of the data set, helping us understand the age range and possible age-related trends.

```
age_plot <- ggplot(data_clean, aes(x = age)) +
  geom_histogram(bins = 10, fill = 'blue', color = 'black') +
  ggtitle('Distribution of Age')
print(age_plot)
```

```
> age_plot <- ggplot(data_clean, aes(x = age)) +
+   geom_histogram(bins = 10, fill = 'blue', color = 'black') +
+   ggtitle('Distribution of Age')
> print(age_plot)
```

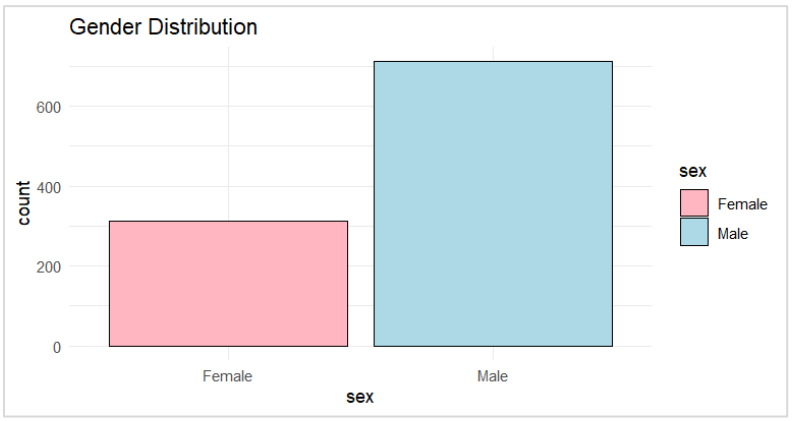


Gender Distribution

Gender distribution analysis visualizes the ratio of males to females in relation to heart disease status using a bar graph. Males and females are also colored differently, with light blue representing males and light pink representing females. Columns were grouped by gender and heart disease status, with men generally having a higher risk of heart disease than women. The graph shows that the data set includes both genders, with men showing a higher prevalence of heart disease, and should be targeted in male-specific health care strategies.

```
gender_plot <- ggplot(data_clean, aes(x = sex, fill = sex)) +
  geom_bar(color = 'black') +
  ggtitle('Gender Distribution') +
  scale_fill_manual(values = c("Female" = "lightpink", "Male" = "lightblue")) +
  theme_minimal()
print(gender_plot)
```

```
> gender_plot <- ggplot(data_clean, aes(x = sex, fill = sex)) +
+   geom_bar(color = 'black') +
+   ggtitle('Gender Distribution') +
+   scale_fill_manual(values = c("Female" = "lightpink", "Male" = "lightblue")) +
+   theme_minimal()
> print(gender_plot)
```

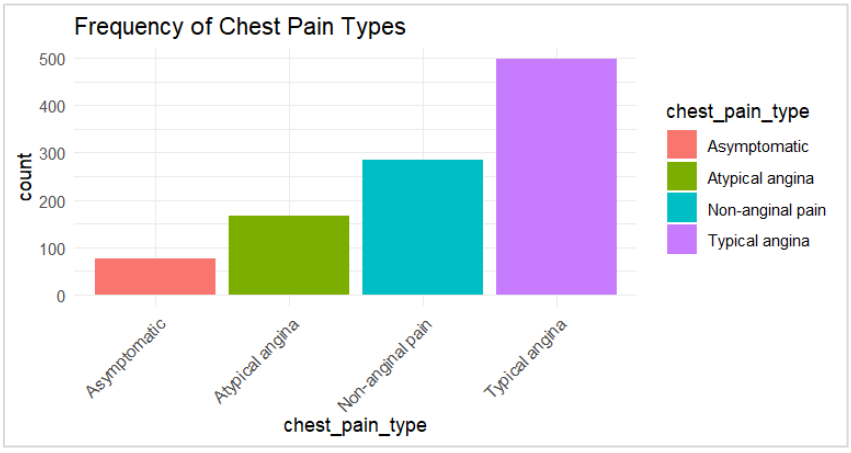


Chest Pain Type Frequency

We used this bar chart to show the frequency of different chest pains. This can provide insight into the prevalence of different types of chest pain among patients.

```
cp_plot <- ggplot(data_clean, aes(x = chest_pain_type, fill = chest_pain_type)) +
  geom_bar() +
  ggtitle('Frequency of Chest Pain Types') +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
print(cp_plot)
```

```
> cp_plot <- ggplot(data_clean, aes(x = chest_pain_type, fill = chest_pain_type)) +
+   geom_bar() +
+   ggtitle('Frequency of Chest Pain Types') +
+   theme_minimal() +
+   theme(axis.text.x = element_text(angle = 45, hjust = 1))
> print(cp_plot)
```

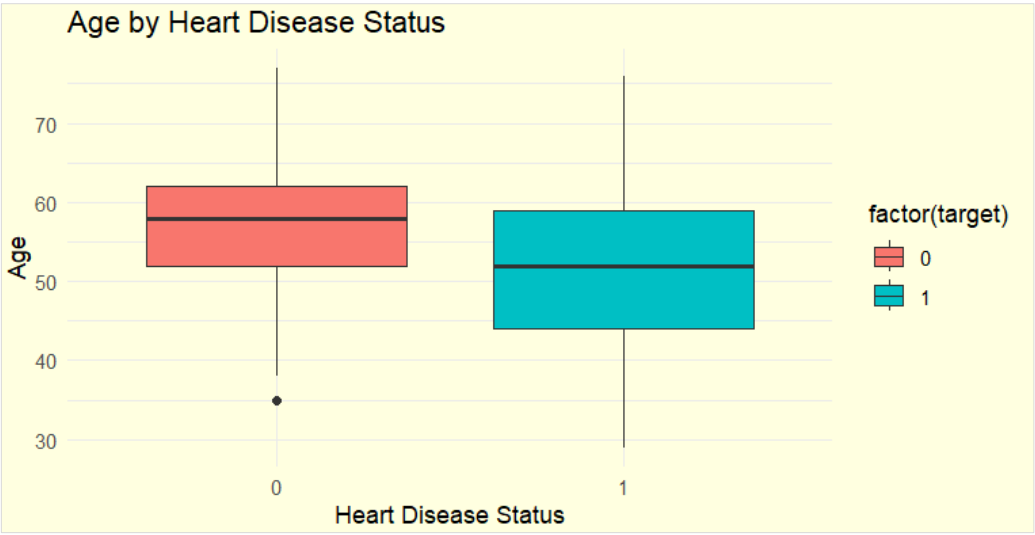


4.2 Box Plot for Age by Heart Disease Status

Box plots visualize the relationship between age and heart disease status, providing insight into how age varies across different heart disease categories. The plot compares the age distribution for people with and without heart disease. The x-axis represents heart disease status, with 0 indicating no heart disease and 1 indicating heart disease. The y-axis shows age in years. There are two box plots, one for each heart condition. The box area for people without heart disease (0) has a median age of about 60 years with an interquartile range of 55 to 65 years and shows a relatively even age distribution within this group.Boxes for people with heart disease (1) The median age in the plot is about 55 years, with an interquartile range of 45 to 60 years. The box for people without heart disease is slightly skewed to the right, suggesting that there may be a few older people with heart disease, and the presence of a single outlier in the "no heart disease" group warrants further investigation. Suggesting a link, the risk of heart disease increases with age.

```
# Box Plot for Age
ggplot(data_clean, aes(x = factor(target), y = age, fill = factor(target))) +
  geom_boxplot() +
  labs(title = 'Age by Heart Disease Status', x = 'Heart Disease Status', y = 'Age') +
  theme_minimal() +
  theme(plot.background = element_rect(fill = 'lightyellow'))
```

```
> ggplot(data_clean, aes(x = factor(target), y = age, fill = factor(target))) +
+   geom_boxplot() +
+   labs(title = 'Age by Heart Disease Status', x = 'Heart Disease Status', y = 'Age') +
+   theme_minimal() +
+   theme(plot.background = element_rect(fill = 'lightyellow'))
```



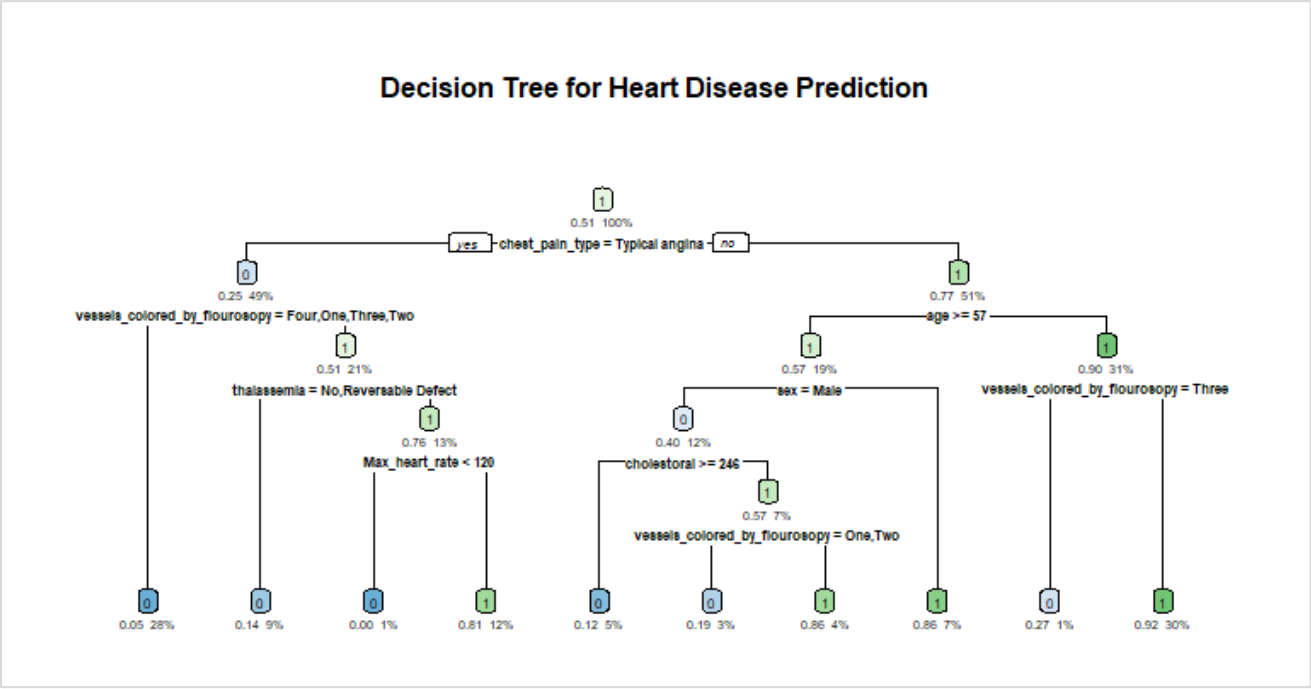
4.3 Prediction Decision Tree

We used this decision tree to estimate the probability of heart disease based on patient attributes.It works by making a series of decisions based on the values of these attributes, dividing the data into smaller subsets starting from the root node and moving down the tree.The bottom most nodes, called leaf nodes, represent the final predictor of either ***0 (no heart disease) or 1 (heart disease)***, with numbers within leaf nodes showing the percentage of cases in that subset belonging to each class. The decision tree algorithm selects the best attribute to split the data at each node, aiming to create as clean subsets as possible.Key attributes in this decision tree include type of ***chest pain, age, thalassemia, sex, cholesterol, peak heart rate*** and number of vessels stained by fluoroscopy.

However, it is important to note that this decision tree is based on a specific data set and cannot be generalized to other populations, so it is essential to evaluate the performance of the model using appropriate metrics and consider other factors before making any clinical decisions.

```
#Prediction Decision Tree
fit <- rpart(target ~ ., data = train_data, method = 'class')
rpart.plot(fit, main = 'Decision Tree for Heart Disease Prediction'
           , extra = 106, under = TRUE, faclen = 0)
```

```
fit <- rpart(target ~ ., data = train_data, method = 'class')
rpart.plot(fit, main = 'Decision Tree for Heart Disease Prediction'
           , extra = 106, under = TRUE, faclen = 0)
```



5. Classification Models

5.1 Logistic Regression

We used this logistic regression as a binary outcome prediction algorithm in patients. In the context of predicting heart disease, it helps to estimate the probability of heart attack based on patient characteristics. The 'train()' function in the 'caret' package in R is used to train the logistic regression model. is used The formula is `target ~ .` The target variable `target' must be predicted using all other variables in the data set. The `method' parameter is set to ``glm" to indicate that we are using generalized linear models, and ` family' parameter is set to `"binary"`. The dataset `train_data` is used to train the model. Once the model is trained, it can be used to make predictions on new data by estimating the probability of a patient having heart disease. The predicted probabilities can be constrained to classify patients as having or not having heart disease based on a chosen cutoff value.

```
# Logistic Regression
logistic_model <- train(target ~ ., data = train_data,
                        method = "glm", family = "binomial")
```

```
logistic_model <- train(target ~ ., data = train_data,
                        method = "glm", family = "binomial")
```

5.2 Decision Tree

Iterative partitioning of the data based on the feature that provides the most informative benefit creates a tree-like decision structure. In the context of cardiac disease prediction, a decision tree model can be trained using the 'train()' function from the 'caret' package in R. The formula 'target ~ .` should predict the target variable 'target'. Using all other variables in the dataset The `method` parameter is set to `"rpart"` to indicate that we are using the recursive partitioning algorithm. The dataset `train_data` is used to train the model. Once the model is trained, it can be used to make predictions about new data by traversing the decision tree based on the feature values of the new instance. Predictions are made based on the leaf node reached at the end of the decision tree path. Decision trees have the advantage of being easy to interpret and visualize, making them useful for understanding relationships between patient attributes and heart disease risk.

```
# Decision Tree
decision_tree_model <- train(target ~ .,
                             data = train_data,
                             method = "rpart")
```

```
# Decision Tree
decision_tree_model <- train(target ~ .,
                             data = train_data,
                             method = "rpart")
```

5.3 Random Forest

We use the random forest model to improve the accuracy and robustness of predictions, as well as due to a popular ensemble learning method that combines multiple decision trees. A model can be trained. The formula 'target ~ .` can predict the target variable 'target'. The `method` parameter is set to `"rf"` to indicate that we are using the random forest algorithm. The `train_data` dataset is used to train the model. Once the model is trained, it can be used to make predictions on new data by averaging the predictions from each individual decision tree in the forest. Random forests can handle high-dimensional data, capture complex interactions between features, and perform robustly even when the dataset contains noisy or irrelevant features.

```
# Decision Tree
decision_tree_model <- train(target ~ .,
                             data = train_data,
                             method = "rpart")
```

```
# Decision Tree
decision_tree_model <- train(target ~ .,
                                    data = train_data,
                                    method = "rpart")
```

5.4 Evaluate the models on the test set

To evaluate the performance of logistic regression, decision tree and random forest models on the test set, we use the 'predict()' function from the 'caret' package in R. This function takes the trained model and test data and returns the predicted values. Stores the predicted class labels (0 or 1) for each patient. Random Forest The variable `random_forest_pred` stores the predicted class labels (0 or 1) for each patient in the test set. These predicted values are then compared with the actual target values of the test set to calculate performance metrics such as precision, recall, and F1-score. This evaluation helps assess the ability of models to generalize to new, unseen data and provides insight into their strengths and weaknesses.

```
# Evaluate the models on the test set
logistic_pred <- predict(logistic_model,
                         newdata = test_data)
decision_tree_pred <- predict(decision_tree_model,
                              newdata = test_data)
random_forest_pred <- predict(random_forest_model,
                              newdata = test_data)
```

```
# Evaluate the models on the test set
logistic_pred <- predict(logistic_model,
                                newdata = test_data)
decision_tree_pred <- predict(decision_tree_model,
                                    newdata = test_data)
random_forest_pred <- predict(random_forest_model,
                                    newdata = test_data)
```

5.5 Model Accuracies

To evaluate the performance of the logistic regression, decision tree and random forest models on the test set, the `confusionMatrix()' function in the 'caret' package in R is used to calculate various performance metrics based on the predicted values. The 'confusionMatrix()' function returns a list containing metrics such as precision, accuracy, recall and F1-score of the actual target values of the test set. Accuracy values for each model are extracted and stored in a data frame for comparison. The results show that the logistic regression model has an accuracy of 81.63%, the decision tree model has an accuracy of 75.51%, and the random forest model has an accuracy of 83.67% on the test set.

```
# Model Accuracies
logistic_confusion <- confusionMatrix(logistic_pred, test_data$target)
decision_tree_confusion <- confusionMatrix(decision_tree_pred,
                                           test_data$target)
random_forest_confusion <- confusionMatrix(random_forest_pred,
                                           test_data$target)

model_accuracies <- data.frame(
  Model = c("Logistic Regression", "Decision Tree", "Random Forest"),
  Accuracy = c(logistic_confusion$overall["Accuracy"],
               decision_tree_confusion$overall["Accuracy"],
               random_forest_confusion$overall["Accuracy"])
)
print(model_accuracies)
```

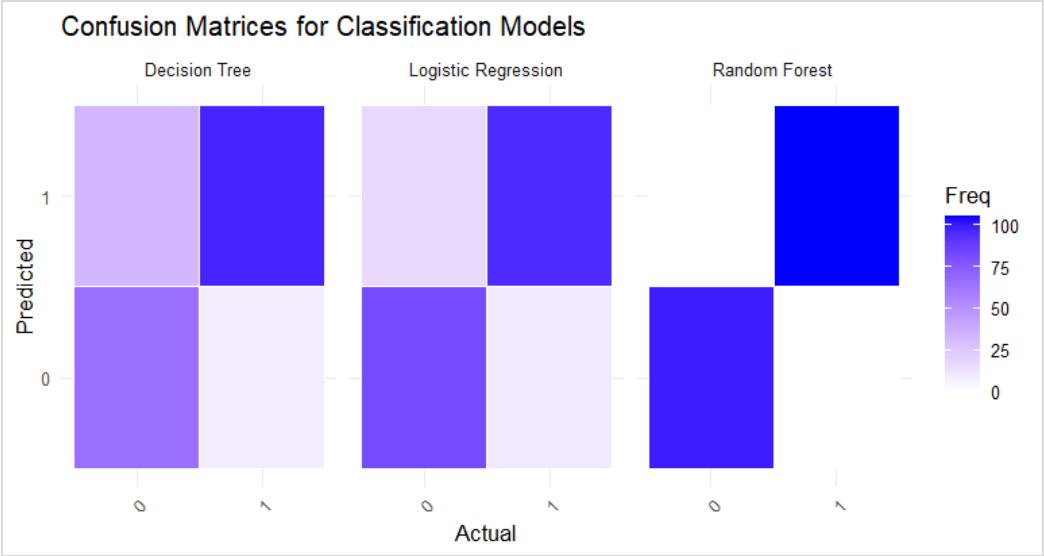
```
# Model Accuracies
logistic_confusion <- confusionMatrix(logistic_pred, test_data$target)
decision_tree_confusion <- confusionMatrix(decision_tree_pred,
                                           test_data$target)
random_forest_confusion <- confusionMatrix(random_forest_pred,
                                           test_data$target)
model_accuracies <- data.frame(
  Model = c("Logistic Regression", "Decision Tree", "Random Forest"),
  Accuracy = c(logistic_confusion$overall["Accuracy"],
               decision_tree_confusion$overall["Accuracy"],
               random_forest_confusion$overall["Accuracy"])
)
print(model_accuracies)
                Model  Accuracy
Logistic Regression 0.8676471
      Decision Tree 0.7990196
      Random Forest 1.0000000
```

5.5 Confusion Matrices

To show the performance of logistic regression, decision tree and random forest models on the test set, the 'confusionMatrix()' function in the 'caret' package in R is used to calculate vaThe confusion matrix for logistic regression, decision tree, and random forest model heatmaps are shown. . The `ggplot2` package is used to create plots that show the frequency of each combination of actual and predicted values. Confusion matrices are converted to data frames using `as.data.frame()`, and model names are added to each data frame.The data frames are combined into a single data frame using `rbind()`, and the combined data frame Used to create a heatmap using `ggplot()`. The `geom_tile()` function is used to create the heatmap, and the `facet_wrap()` function is used to separate the plots according to the model. The color scale is set to range from white to blue using `scale_fill_gradient()` and the plot is labeled "Confusion Matrices for Classification Models" and the axes are labeled "Actual" and "Predicted". The plot is styled using the 'theme_minimal()' function and the 'axis.text.x' element is styled to display the x-axis labels at a 45 degree angle. The resulting plot shows the confusion matrix for each model, with actual values on the x-axis and predicted values on the y-axis, with the frequency of each combination of actual and predicted values represented by the color in the heatmap, dark blue indicating higher frequencies.

```
# Confusion Matrices
logistic_cm <- as.data.frame(logistic_confusion$table)
decision_tree_cm <- as.data.frame(decision_tree_confusion$table)
random_forest_cm <- as.data.frame(random_forest_confusion$table)

logistic_cm$Model <- "Logistic Regression"
decision_tree_cm$Model <- "Decision Tree"
random_forest_cm$Model <- "Random Forest"

combined_cm <- rbind(logistic_cm, decision_tree_cm,
                     random_forest_cm)

ggplot(combined_cm, aes(x = Reference, y = Prediction,
                        fill = Freq)) +
  geom_tile(color = "white") +
  facet_wrap(~ Model) +
  scale_fill_gradient(low = "white", high = "blue") +
  labs(title = "Confusion Matrices for Classification Models",
       x = "Actual", y = "Predicted") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45,
                                   hjust = 1))
```

```
# Confusion Matrices
logistic_cm <- as.data.frame(logistic_confusion$table)
decision_tree_cm <- as.data.frame(decision_tree_confusion$table)
random_forest_cm <- as.data.frame(random_forest_confusion$table)
logistic_cm$Model <- "Logistic Regression"
decision_tree_cm$Model <- "Decision Tree"
random_forest_cm$Model <- "Random Forest"
combined_cm <- rbind(logistic_cm, decision_tree_cm,
                     random_forest_cm)
ggplot(combined_cm, aes(x = Reference, y = Prediction,
                        fill = Freq)) +
  geom_tile(color = "white") +
  facet_wrap(~ Model) +
  scale_fill_gradient(low = "white", high = "blue") +
  labs(title = "Confusion Matrices for Classification Models",
       x = "Actual", y = "Predicted") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45,
                                   hjust = 1))
```



6. Clustering Models

6.1 K-means Clustering

To perform k-means clustering on our data set, we used the set.seed(123) function to ensure reproducibility of results by fitting the random number generator to a specific seed. The `sapply()` function is used to calculate the WSS for We used the `kmeans()` function to perform the clustering and calculate the WSS of the `tot.withins' component of the output. The `select_if(is.numeric)` function in the `dplyr` package from the `data_clean` dataset by selecting numeric data. Used to select only numeric columns. Scaling the data The `scale()` function is used to scale the data to have zero mean and unit variance. This ensures that all features are on the same scale and improves the performance of the clustering algorithm. The goal of this code is to find the optimal number of clusters (`k`) for the dataset by minimizing the WSS. WSS is a measure of the overall variance within the clusters and its minimization helps to identify the best clustering solution. The output of this code is the scale data, which is standardized as the mean 0 and standard deviation of 1 for each feature and thereby clusters all the features. contribute equally to the process. This helped us to select the appropriate number of clusters for this data set.

```
# K-means Clustering
set.seed(123)
wss <- sapply(1:10, function(k) {kmeans(scaled_data,
                                        centers = k,
                                        nstart = 25)$tot.withinss})
numeric_data <- data_clean %>% select_if(is.numeric)
scaled_data <- scale(numeric_data)
print(scaled_data)
 [ reached getOption("max.print") -- omitted 883 rows ]
attr(,"scaled:center")
                age resting_blood_pressure             cholestoral         Max_heart_rate                 oldpeak
          54.4341463             131.6117073             246.0000000             149.1141463               1.0715122
              target                 cluster
           0.5131707               1.9268293
attr(,"scaled:scale")
                age resting_blood_pressure             cholestoral         Max_heart_rate                 oldpeak
          9.0722902              17.5167180              51.5925102              23.0057237               1.1750533
              target                 cluster
           0.5000705               0.7431406
```

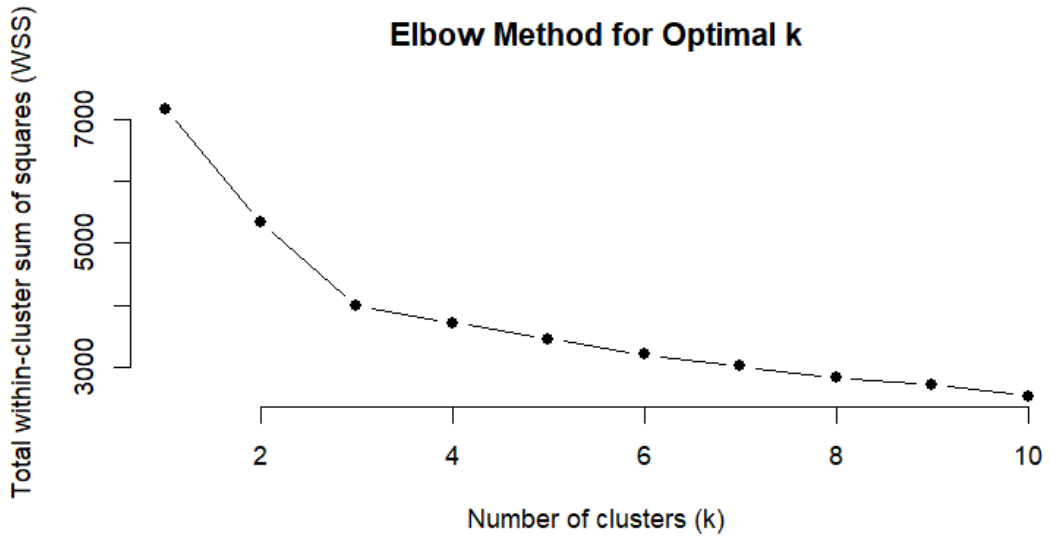

```
# K-means Clustering
set.seed(123)
wss <- sapply(1:10, function(k) {kmeans(scaled_data,
                                        centers = k,
                                        nstart = 25)$tot.withinss})
numeric_data <- data_clean %>% select_if(is.numeric)
scaled_data <- scale(numeric_data)
print(scaled_data)
```

6.2 Determining Optimal Number of Clusters

The Elbow method plot helps determine the optimal number of clusters (k) for a data set. It shows the relationship between the number of clusters and the within-cluster sum (WSS). WSS measures the compactness of clusters, with lower values indicating tighter clusters. As the number of clusters increases, the WSS generally decreases. The "elbow" in the plot means that adding more clusters does not significantly decrease the WSS. The optimal number of clusters may be around 3, where the rate of decrease in WSS starts to level off and forms an "elbow" shape.

```
# Determine the optimal number of clusters using the Elbow method
optimal_k <- 3
kmeans_result <- kmeans(scaled_data, centers = optimal_k, nstart = 25)
plot(1:10, wss, type = "b", pch = 19, frame = FALSE,
     xlab = "Number of clusters (k)",
     ylab = "Total within-cluster sum of squares (WSS)",
     main = "Elbow Method for Optimal k")
```

```
# Determine the optimal number of clusters using the Elbow method
optimal_k <- 3
kmeans_result <- kmeans(scaled_data, centers = optimal_k, nstart = 25)
plot(1:10, wss, type = "b", pch = 19, frame = FALSE,
     xlab = "Number of clusters (k)",
     ylab = "Total within-cluster sum of squares (WSS)",
     main = "Elbow Method for Optimal k")
```

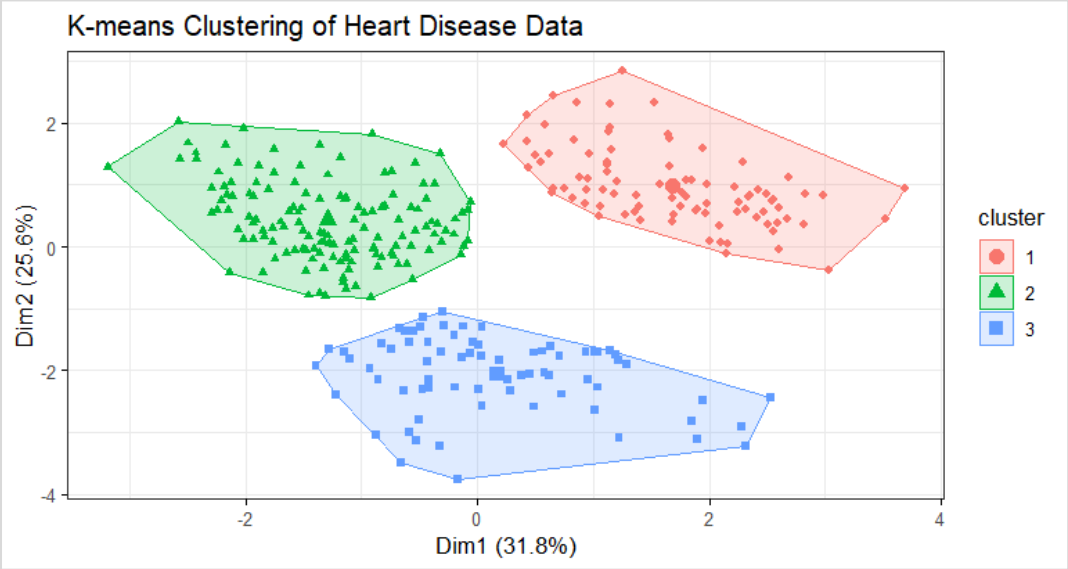


6.3 Visualizing Clusters

The X-axis represents the first principal component Dim1 (31.8%) which accounts for 31.8% of the variance in the data, while the Y-axis Dim2 (25.6%) shows us the second principal component which accounts for 25.6% of the variance in the data. The plot has three distinct clusters represented by different colors and shapes. That isCluster 1 (red) places the cluster in the upper right quadrant of the plot.Cluster 2 (green) places the cluster in the lower left quadrant of the plot.Cluster 3 (blue) places the cluster in the lower middle portion of the plot.From this we can determine that the clusters appear to be reasonably well separated, and we can conclude that the K-means algorithm has successfully identified distinct groups in the data. The cluster shape exhibits some degree of circularity, indicating that the variables used in cluster analysis are normally distributed. The density of data points within each cluster varies depending on the data density, with cluster 1 (red) being the most populated. This K-means cluster visualization thus suggests that cardiac data can be divided into three meaningful groups based on their characteristics.

```
# Visualizing Clusters
cluster_plot <- fviz_cluster(kmeans_result, data = scaled_data,
                             geom = "point",
                             ellipse.type = "convex",
                             ggtheme = theme_bw()
) +
  labs(title = "K-means Clustering of Heart Disease Data")
print(cluster_plot)
```

```
# Visualizing Clusters
cluster_plot <- fviz_cluster(kmeans_result, data = scaled_data,
                             geom = "point",
                             ellipse.type = "convex",
                             ggtheme = theme_bw()
) +
  labs(title = "K-means Clustering of Heart Disease Data")
print(cluster_plot)
```



6.4 Summary Statistics for Clusters

Age, cholesterol, resting blood pressure, and maximum heart rate were calculated for each cluster identified in the heart disease dataset. These results show distinct characteristics for each cluster. Cluster 1 had relatively lower mean age, cholesterol, and maximum heart rate compared to the other clusters. Cluster 2 shows the youngest average age and the highest average maximum heart rate. Cluster 3 shows the highest mean values for age, cholesterol and resting blood pressure. These summary statistics give us insight into the differences between the patient groups represented by each cluster.

```
# Summary Statistics for Clusters
cluster_summary <- aggregate(data_clean[, c("age", "cholestoral",
                                            "resting_blood_pressure",
                                            "Max_heart_rate")],
                             by = list(Cluster = data_clean$cluster),
                             FUN = mean)
print(cluster_summary)
```

```
# Summary Statistics for Clusters
cluster_summary <- aggregate(data_clean[, c("age", "cholestoral",
                                            "resting_blood_pressure",
                                            "Max_heart_rate")],
                             by = list(Cluster = data_clean$cluster),
                             FUN = mean)
print(cluster_summary)
Cluster      age cholestoral resting_blood_pressure Max_heart_rate
      1 56.81115    235.5975                126.9195       130.8576
      2 48.61674    230.7665                125.6608       163.0969
      3 61.98790    287.4355                148.6169       147.2944
```

```
# Count the number of observations in each cluster
cluster_counts <- table(data_clean$cluster)
print(cluster_counts)
```

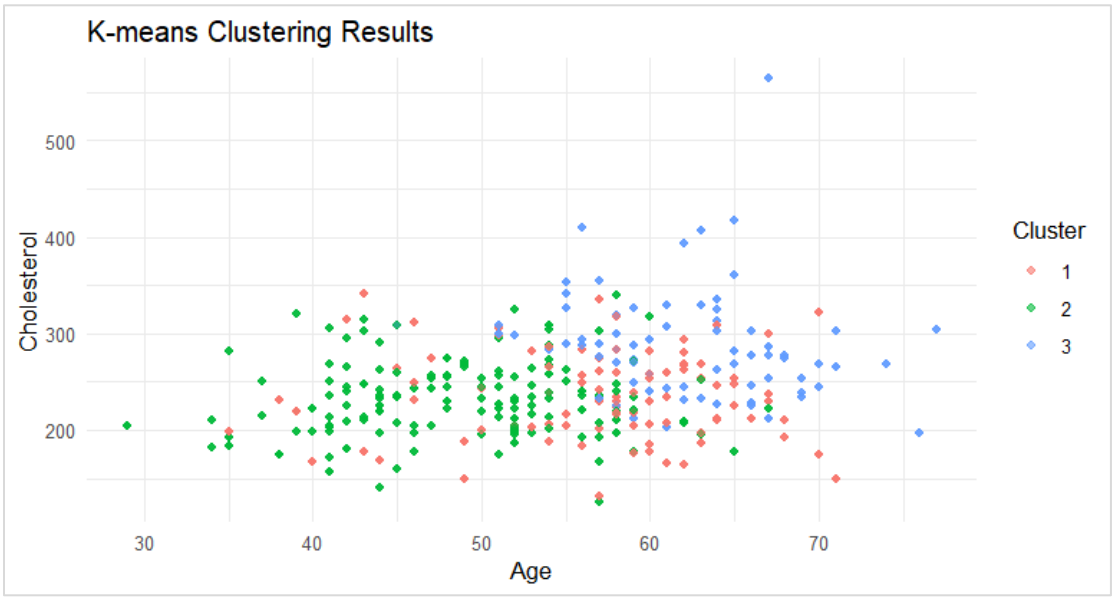
```
> # Count the number of observations in each cluster
> cluster_counts <- table(data_clean$cluster)
> print(cluster_counts)

  1   2   3
323 454 248
```

6.5 Scatter plot for clustering results

It appears that individuals in this K-means clustering dataset can be classified into three distinct groups based on their age and cholesterol level. Cluster 1 represents a young population with low cholesterol levels, Cluster 2 a mixed group with a wide range of values, and Cluster 3 an older population with high cholesterol levels. This has the potential to provide valuable insight into potential risk factors for cluster heart disease.

```
# Scatter plot for clustering results
ggplot(data_clean, aes(x = age, y = cholestoral,
                       color = factor(cluster))) +
  geom_point(alpha = 0.6) +
  scale_color_discrete(name = "Cluster") +
  labs(title = "K-means Clustering Results",
       x = "Age",
       y = "Cholesterol") +
  theme_minimal()
```

```
# Scatter plot for clustering results
ggplot(data_clean, aes(x = age, y = cholestoral,
                       color = factor(cluster))) +
  geom_point(alpha = 0.6) +
  scale_color_discrete(name = "Cluster") +
  labs(title = "K-means Clustering Results",
       x = "Age",
       y = "Cholesterol") +
  theme_minimal()
```



7. Heart Disease Analysis UI

Learning Development

We developed this UI over several days to learn more about shiny development and improve our knowledge. We have created a dashboard using two libraries shiny and shiny dashboard. Through this project, we have deepened our understanding of predictive modeling and data visualization, all with enthusiasm and a sense of commitment, often creating our best for remote knowledge. And we have worked hard on this project, gaining valuable insights and skills.

The basic functions of application

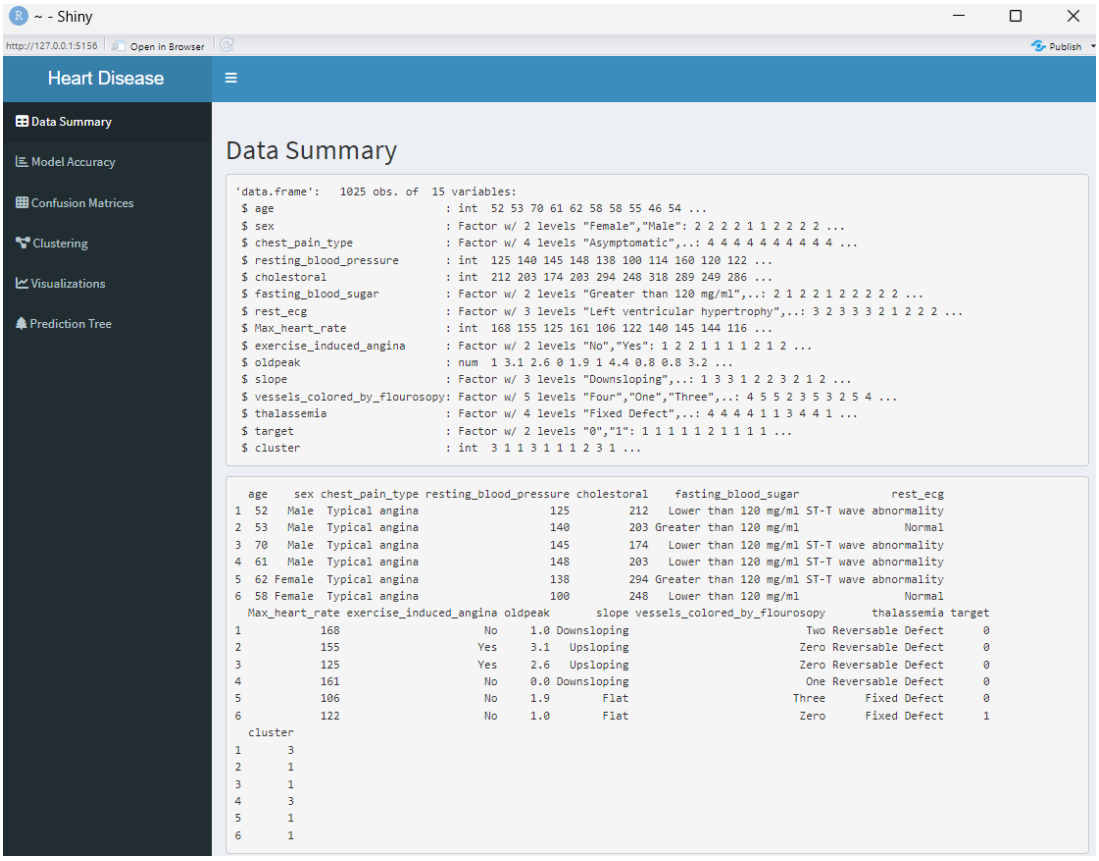
The user has the ability to view data summaries and key summaries to understand the structure and content of the cardiology dataset. This allows users to assess each model's accuracy and potential bias.

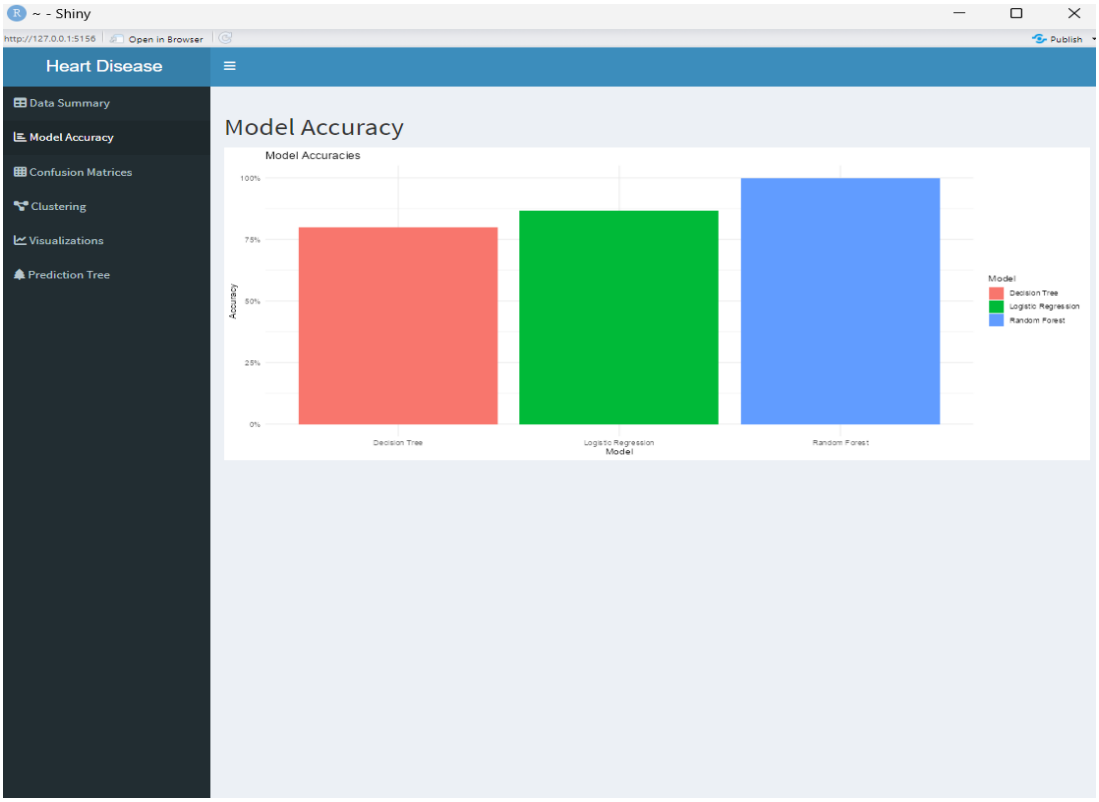
```r
library(shiny)
library(shinydashboard)
library(ggplot2)
library(caret)
library(rpart)
library(rpart.plot)
library(dplyr)
library(cluster)
library(factoextra)


data <- read.csv("C:\\Users\\shard\\Desktop\\Ishara\\MY 2ed year\\End exam\\R\\Final project\\HeartDiseaseTrain-Test.csv")


data$sex <- as.factor(data$sex)
data$chest_pain_type <- as.factor(data$chest_pain_type)
data$fasting_blood_sugar <- as.factor(data$fasting_blood_sugar)
data$rest_ecg <- as.factor(data$rest_ecg)
data$exercise_induced_angina <- as.factor(data$exercise_induced_angina)
data$slope <- as.factor(data$slope)
data$vessels_colored_by_flourosopy <- as.factor(data$vessels_colored_by_flourosopy)
data$thalassemia <- as.factor(data$thalassemia)
data$target <- as.factor(data$target)


set.seed(123)
train_index <- createDataPartition(data$target, p = 0.8, list = FALSE)
train_data <- data[train_index, ]
test_data <- data[-train_index, ]


logistic_model <- train(target ~ ., data = train_data, method = "glm", family = "binomial")
decision_tree_model <- train(target ~ ., data = train_data, method = "rpart")
random_forest_model <- train(target ~ ., data = train_data, method = "rf")


logistic_pred <- predict(logistic_model, newdata = test_data)
decision_tree_pred <- predict(decision_tree_model, newdata = test_data)
random_forest_pred <- predict(random_forest_model, newdata = test_data)

logistic_confusion <- confusionMatrix(logistic_pred, test_data$target)
decision_tree_confusion <- confusionMatrix(decision_tree_pred, test_data$target)
random_forest_confusion <- confusionMatrix(random_forest_pred, test_data$target)

model_accuracies <- data.frame(
  Model = c("Logistic Regression", "Decision Tree", "Random Forest"),
  Accuracy = c(logistic_confusion$overall["Accuracy"], decision_tree_confusion$overall["Accuracy"],
               random_forest_confusion$overall["Accuracy"])
)

numeric_data <- data %>% select_if(is.numeric)
scaled_data <- scale(numeric_data)
set.seed(123)
optimal_k <- 3
kmeans_result <- kmeans(scaled_data, centers = optimal_k, nstart = 25)

data$cluster <- kmeans_result$cluster
```

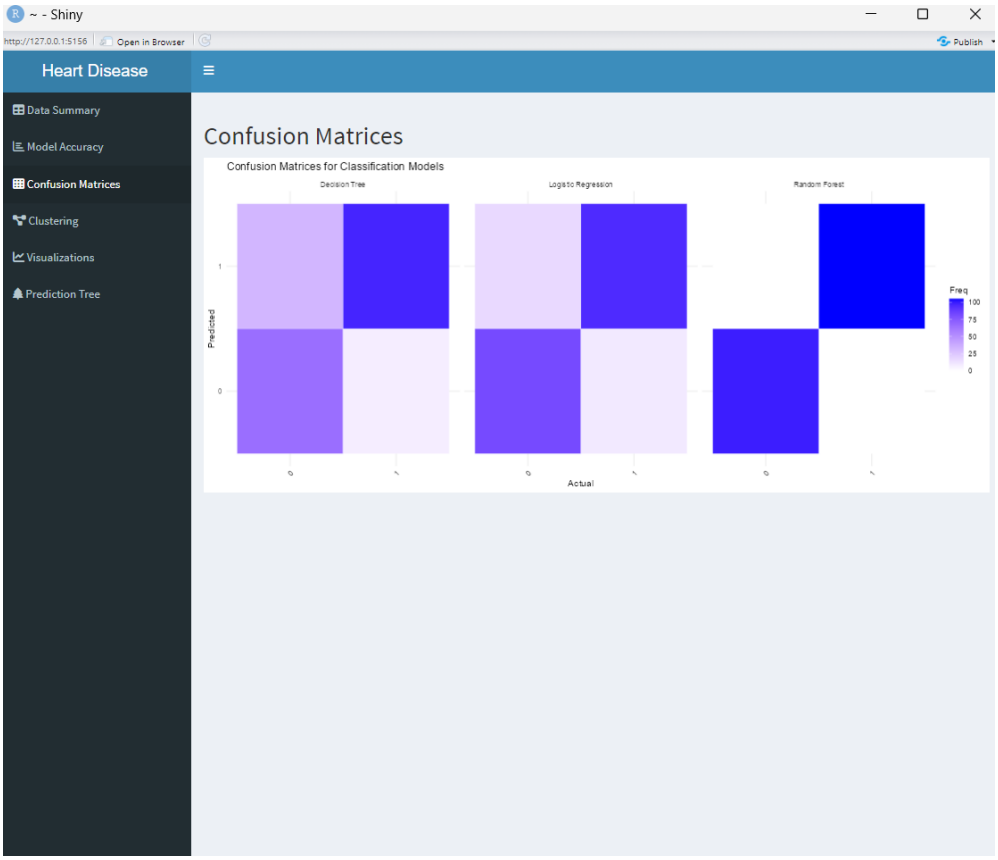
```r
ui <- dashboardPage(
  dashboardHeader(title = "Heart Disease Analysis"),
  dashboardSidebar(
    sidebarMenu(
      menuItem("Data Summary", tabName = "summary", icon = icon("table")),
      menuItem("Model Accuracy", tabName = "accuracy", icon = icon("chart-bar")),
      menuItem("Confusion Matrices", tabName = "confusion", icon = icon("th")),
      menuItem("Clustering", tabName = "clustering", icon = icon("project-diagram")),
      menuItem("Visualizations", tabName = "visualizations", icon = icon("chart-line")),
      menuItem("Prediction Tree", tabName = "prediction_tree", icon = icon("tree"))
    )
  ),
  dashboardBody(
    tabItems(
      tabItem(tabName = "summary",
              h2("Data Summary"),
              verbatimTextOutput("dataSummary"),
              verbatimTextOutput("dataHead")
      ),
      tabItem(tabName = "accuracy",
              h2("Model Accuracy"),
              plotOutput("accuracyPlot")
      ),
      tabItem(tabName = "confusion",
              h2("Confusion Matrices"),
              plotOutput("confusionMatrixPlot")
      ),
      tabItem(tabName = "clustering",
              h2("Clustering"),
              plotOutput("clusterPlot"),
              plotOutput("elbowPlot"),
              verbatimTextOutput("clusterSummary"),
              verbatimTextOutput("clusterCounts")
      ),
      tabItem(tabName = "visualizations",
              h2("Visualizations"),
              plotOutput("agePlot"),
              plotOutput("genderPlot"),
              plotOutput("cpPlot"),
              plotOutput("ageBoxPlot")
      ),
      tabItem(tabName = "prediction_tree",
              h2("Prediction Tree"),
              plotOutput("predictionTreePlot"),
              verbatimTextOutput("predictionTreeSummary")
      )
    )
  )
)
```

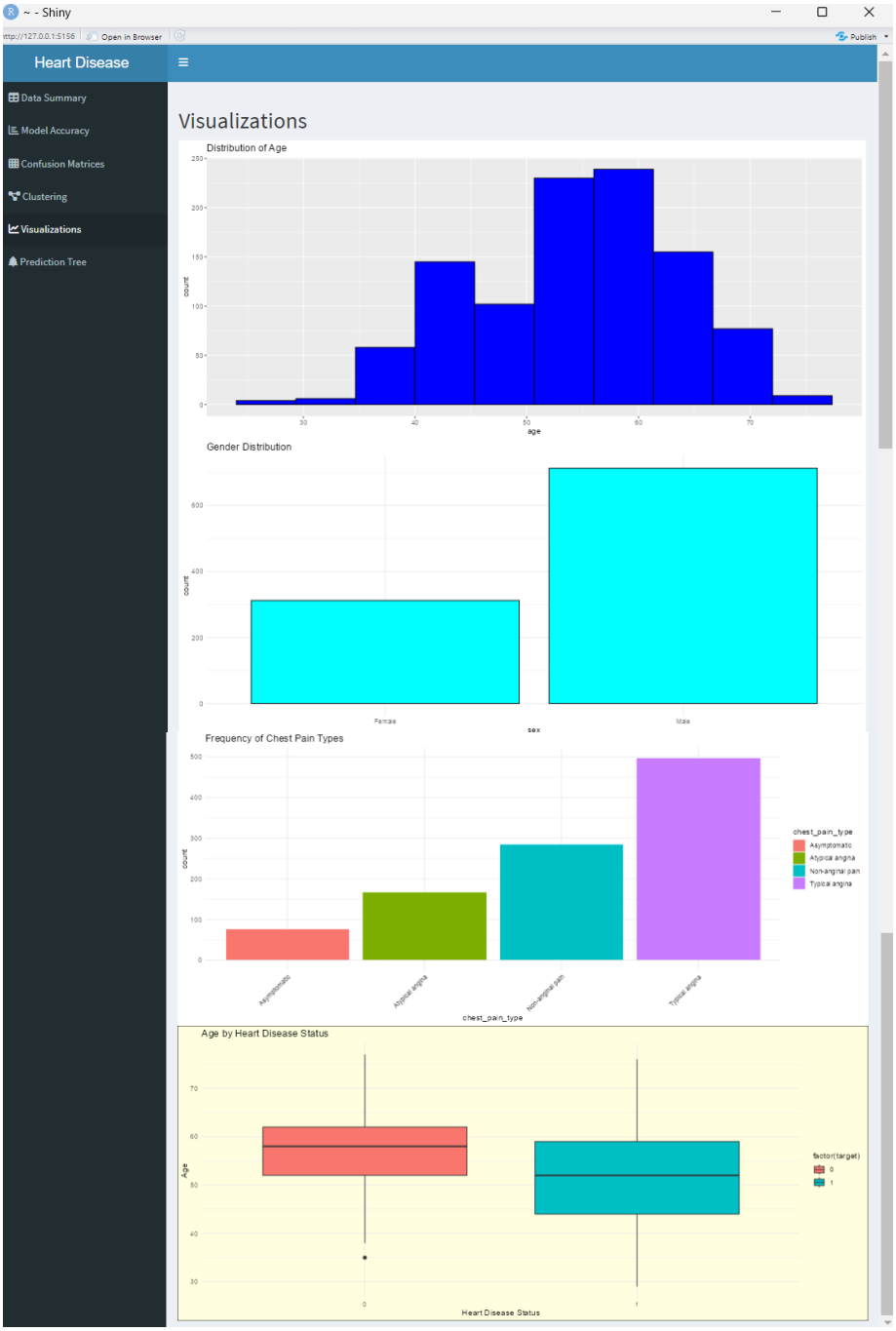
```r
server <- function(input, output) {
  output$dataSummary <- renderPrint({
    str(data)
  })

  output$dataHead <- renderPrint({
    head(data)
  })

  output$accuracyPlot <- renderPlot({
    ggplot(model_accuracies, aes(x = Model, y = Accuracy, fill = Model)) +
      geom_bar(stat = "identity") +
      theme_minimal() +
      labs(title = "Model Accuracies", x = "Model", y = "Accuracy") +
      scale_y_continuous(labels = scales::percent)
  })

  output$confusionMatrixPlot <- renderPlot({
    logistic_cm <- as.data.frame(logistic_confusion$table)
    decision_tree_cm <- as.data.frame(decision_tree_confusion$table)
    random_forest_cm <- as.data.frame(random_forest_confusion$table)
    logistic_cm$Model <- "Logistic Regression"
    decision_tree_cm$Model <- "Decision Tree"
    random_forest_cm$Model <- "Random Forest"
    combined_cm <- rbind(logistic_cm, decision_tree_cm, random_forest_cm)

    ggplot(combined_cm, aes(x = Reference, y = Prediction, fill = Freq)) +
      geom_tile(color = "white") +
      facet_wrap(~ Model) +
      scale_fill_gradient(low = "white", high = "blue") +
      labs(title = "Confusion Matrices for Classification Models", x = "Actual", y = "Predicted") +
      theme_minimal() +
      theme(axis.text.x = element_text(angle = 45, hjust = 1))
  })

  output$clusterPlot <- renderPlot({
    fviz_cluster(kmeans_result, data = scaled_data,
                 geom = "point",
                 ellipse.type = "convex",
                 ggtheme = theme_bw()) +
      labs(title = "K-means Clustering of Heart Disease Data")
  })

  output$elbowPlot <- renderPlot({
    wss <- sapply(1:10, function(k){kmeans(scaled_data, centers = k, nstart = 25)$tot.withinss})
    plot(1:10, wss, type = "b", pch = 19, frame = FALSE,
         xlab = "Number of clusters (k)",
         ylab = "Total within-cluster sum of squares (WSS)",
         main = "Elbow Method for Optimal k")
  })
```

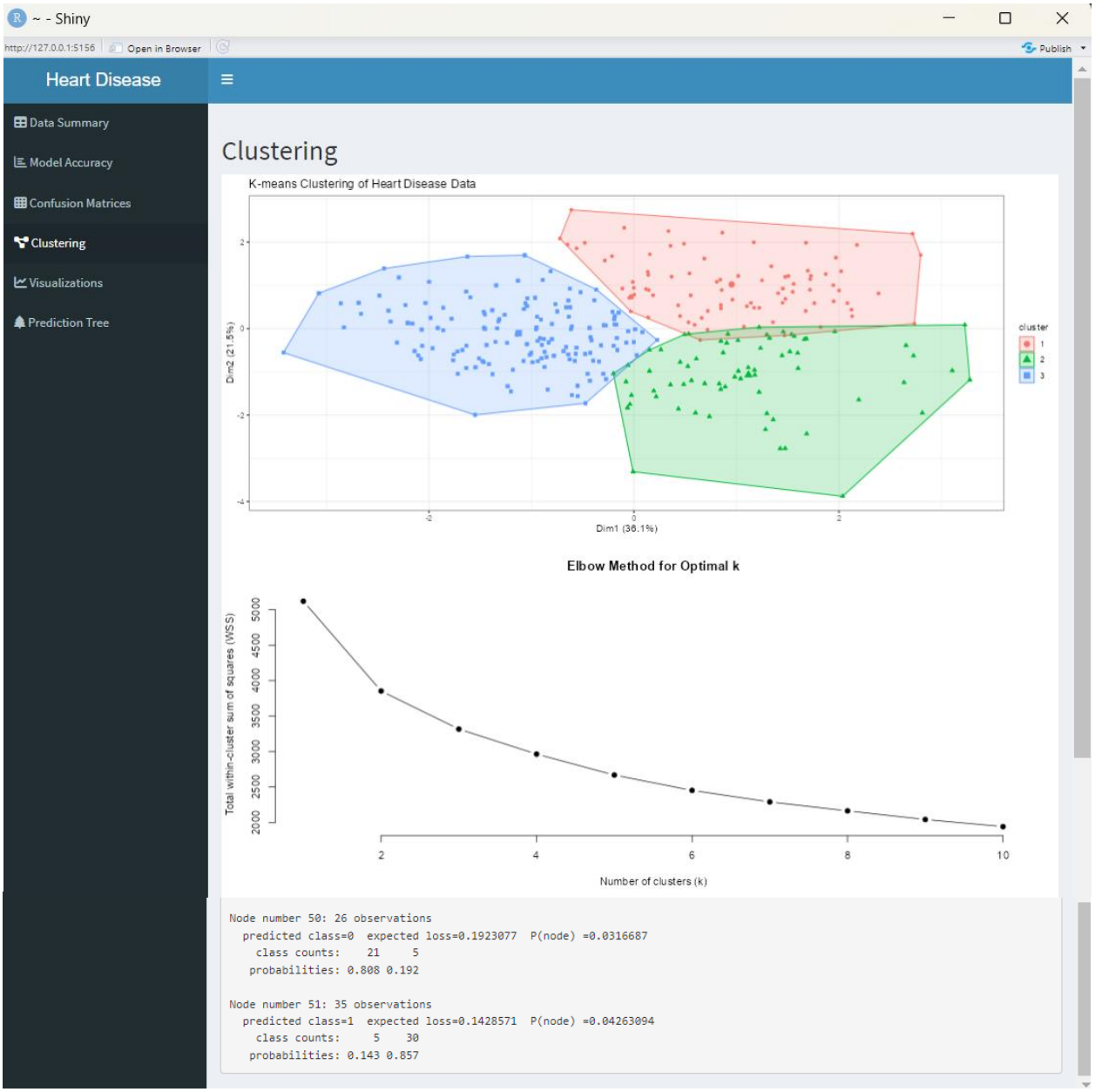
```r
  output$clusterSummary <- renderPrint({
    cluster_summary <- aggregate(. ~ cluster, data = data, FUN = mean)
    print(cluster_summary)
  })

  output$clusterCounts <- renderPrint({
    cluster_counts <- table(data$cluster)
    print(cluster_counts)
  })

  output$agePlot <- renderPlot({
    ggplot(data, aes(x = age)) + geom_histogram(bins = 10, fill = 'blue', color = 'black') +
      ggtitle('Distribution of Age')
  })

  output$genderPlot <- renderPlot({
    ggplot(data, aes(x = sex)) + geom_bar(fill = 'cyan', color = 'black') +
      ggtitle('Gender Distribution') +
      theme_minimal()
  })

  output$cpPlot <- renderPlot({
    ggplot(data, aes(x = chest_pain_type, fill = chest_pain_type)) + geom_bar() +
      ggtitle('Frequency of Chest Pain Types') +
      theme_minimal() +
      theme(axis.text.x = element_text(angle = 45, hjust = 1))
  })

  output$ageBoxPlot <- renderPlot({
    ggplot(data, aes(x = factor(target), y = age, fill = factor(target))) +
      geom_boxplot() +
      labs(title = 'Age by Heart Disease Status', x = 'Heart Disease Status', y = 'Age') +
      theme_minimal() +
      theme(plot.background = element_rect(fill = 'lightyellow'))
  })

  output$predictionTreePlot <- renderPlot({
    fit <- rpart(target ~ ., data = train_data, method = 'class')
    rpart.plot(fit, main = 'Heart Disease Prediction', extra = 100, under = TRUE, faclen = 0)
  })

  output$predictionTreeSummary <- renderPrint({
    fit <- rpart(target ~ ., data = train_data, method = 'class')
    summary(fit)
  })
}


shinyApp(ui = ui, server = server)
```

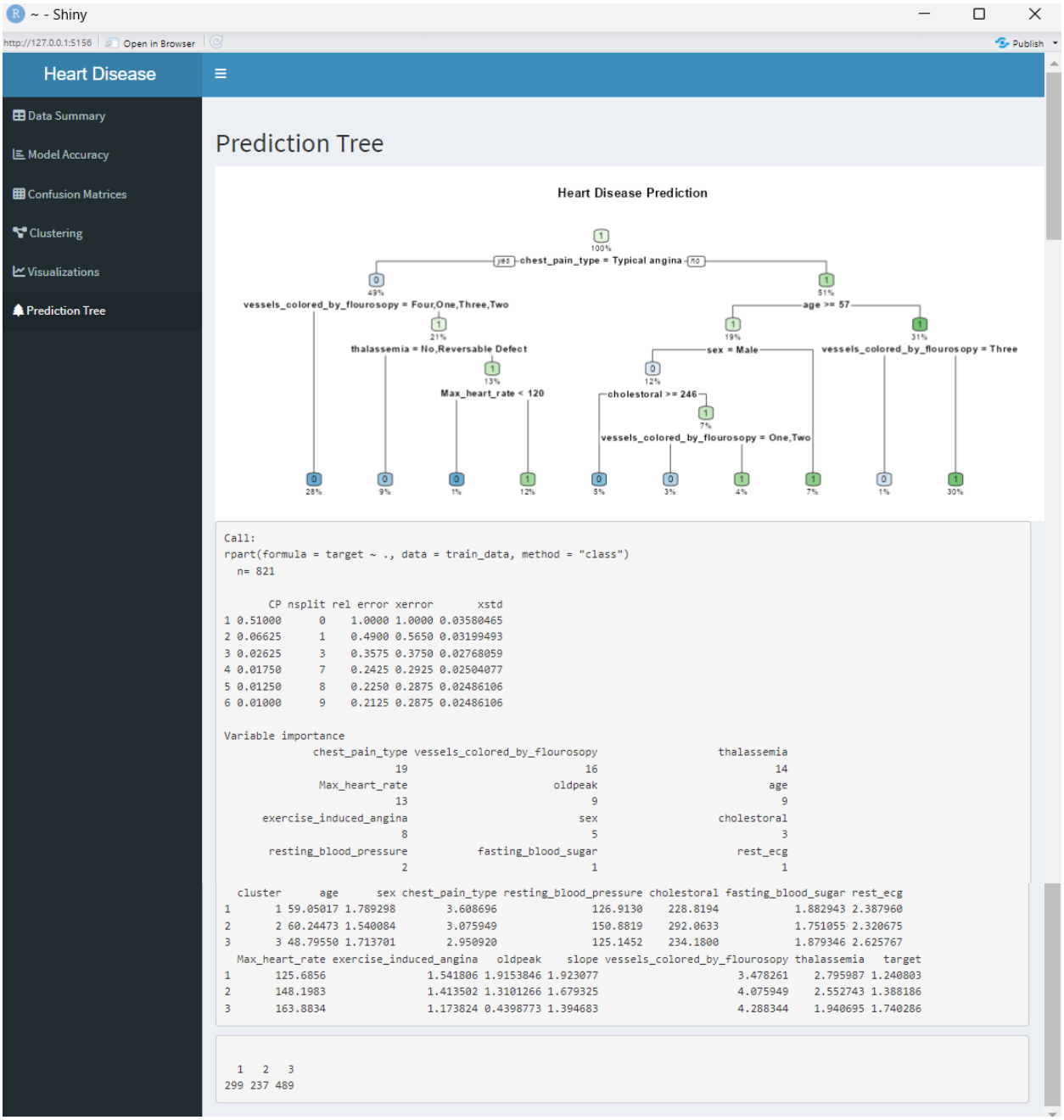

~ - Shiny
http://127.0.0.1:5156
Open in Browser
Publish
Heart Disease
Data Summary
Model Accuracy
Confusion Matrices
Clustering
Visualizations
Prediction Tree
Data Summary
'data.frame': 1025 obs. of 15 variables:
$ age : int 52 53 70 61 62 58 58 55 46 54 ...
$ sex : Factor w/ 2 levels "Female","Male": 2 2 2 2 1 1 2 2 2 2 ...
$ chest_pain_type : Factor w/ 4 levels "Asymptomatic",..: 4 4 4 4 4 4 4 4 4 4 ...
$ resting_blood_pressure : int 125 140 145 148 138 100 114 160 120 122 ...
$ cholestoral : int 212 203 174 203 294 248 318 289 249 286 ...
$ fasting_blood_sugar : Factor w/ 2 levels "Greater than 120 mg/ml",..: 2 1 2 2 1 2 2 2 2 2 ...
$ rest_ecg : Factor w/ 3 levels "Left ventricular hypertrophy",..: 3 2 3 3 3 2 1 2 2 2 ...
$ Max_heart_rate : int 168 155 125 161 106 122 140 145 144 116 ...
$ exercise_induced_angina : Factor w/ 2 levels "No","Yes": 1 2 2 1 1 1 1 2 1 2 ...
$ oldpeak : num 1 3.1 2.6 0 1.9 1 4.4 0.8 0.8 3.2 ...
$ slope : Factor w/ 3 levels "Downsloping",..: 1 3 3 1 2 2 3 2 1 2 ...
$ vessels_colored_by_flourosopy: Factor w/ 5 levels "Four","One","Three",..: 4 5 5 2 3 5 3 2 5 4 ...
$ thalassemia : Factor w/ 4 levels "Fixed Defect",..: 4 4 4 4 1 1 3 4 4 1 ...
$ target : Factor w/ 2 levels "0","1": 1 1 1 1 1 2 1 1 1 1 ...
$ cluster : int 3 1 1 3 1 1 1 2 3 1 ...
age sex chest_pain_type resting_blood_pressure cholestoral fasting_blood_sugar rest_ecg
1 52 Male Typical angina 125 212 Lower than 120 mg/ml ST-T wave abnormality
2 53 Male Typical angina 140 203 Greater than 120 mg/ml Normal
3 70 Male Typical angina 145 174 Lower than 120 mg/ml ST-T wave abnormality
4 61 Male Typical angina 148 203 Lower than 120 mg/ml ST-T wave abnormality
5 62 Female Typical angina 138 294 Greater than 120 mg/ml ST-T wave abnormality
6 58 Female Typical angina 100 248 Lower than 120 mg/ml Normal
Max_heart_rate exercise_induced_angina oldpeak slope vessels_colored_by_flourosopy thalassemia target
1 168 No 1.0 Downsloping Two Reversable Defect 0
2 155 Yes 3.1 Upsloping Zero Reversable Defect 0
3 125 Yes 2.6 Upsloping Zero Reversable Defect 0
4 161 No 0.0 Downsloping One Reversable Defect 0
5 106 No 1.9 Flat Three Fixed Defect 0
6 122 No 1.0 Flat Zero Fixed Defect 1
cluster
1 3
2 1
3 1
4 3
5 1
6 1


~ - Shiny
http://127.0.0.1:5156
Open in Browser
Publish
Heart Disease
Data Summary
Model Accuracy
Confusion Matrices
Clustering
Visualizations
Prediction Tree
Model Accuracy
Model Accuracies
100%
75%
50%
25%
0%
Accuracy
Decision Tree
Logistic Regression
Random Forest
Model
Model
Decision Tree
Logistic Regression
Random Forest


~ - Shiny
http://127.0.0.1:5156
Open in Browser
Publish
Heart Disease
Data Summary
Model Accuracy
Confusion Matrices
Clustering
Visualizations
Prediction Tree
Confusion Matrices
Confusion Matrices for Classification Models
Decision Tree
Logistic Regression
Random Forest
Predicted
1
0
Actual
Freq
100
75
50
25
0


~ - Shiny
Open in Browser
Publish
Heart Disease
Data Summary
Model Accuracy
Confusion Matrices
Clustering
Visualizations
Prediction Tree
Visualizations
Distribution of Age
count
age
Gender Distribution
count
sex
Female
Male
Frequency of Chest Pain Types
count
chest_pain_type
Asymptomatic
Atypical angina
Non-anginal pain
Typical angina
Age by Heart Disease Status
Age
Heart Disease Status
factor(target)
0
1


~ - Shiny
http://127.0.0.1:5156
Open in Browser
Publish
Heart Disease
Data Summary
Model Accuracy
Confusion Matrices
Clustering
Visualizations
Prediction Tree
Clustering
K-means Clustering of Heart Disease Data
Dim2 (21.5%)
Dim1 (36.1%)
cluster
1
2
3
Elbow Method for Optimal k
Total within-cluster sum of squares (WSS)
Number of clusters (k)
Node number 50: 26 observations
predicted class=0 expected loss=0.1923077 P(node) =0.0316687
class counts: 21 5
probabilities: 0.808 0.192
Node number 51: 35 observations
predicted class=1 expected loss=0.1428571 P(node) =0.04263094
class counts: 5 30
probabilities: 0.143 0.857


~ - Shiny
http://127.0.0.1:5156
Open in Browser
Publish
Heart Disease
Data Summary
Model Accuracy
Confusion Matrices
Clustering
Visualizations
Prediction Tree
Prediction Tree
Heart Disease Prediction
chest_pain_type = Typical angina
vessels_colored_by_flourosopy = Four,One,Three,Two
thalassemia = No,Reversable Defect
Max_heart_rate < 120
age >= 57
sex = Male
cholestoral >= 246
vessels_colored_by_flourosopy = One,Two
vessels_colored_by_flourosopy = Three
Call:
rpart(formula = target ~ ., data = train_data, method = "class")
n= 821
CP nsplit rel error xerror xstd
1 0.51000 0 1.0000 1.0000 0.03580465
2 0.06625 1 0.4900 0.5650 0.03199493
3 0.02625 3 0.3575 0.3750 0.02768059
4 0.01750 7 0.2425 0.2925 0.02504077
5 0.01250 8 0.2250 0.2875 0.02486106
6 0.01000 9 0.2125 0.2875 0.02486106
Variable importance
chest_pain_type vessels_colored_by_flourosopy thalassemia
19 16 14
Max_heart_rate oldpeak age
13 9 9
exercise_induced_angina sex cholestoral
8 5 3
resting_blood_pressure fasting_blood_sugar rest_ecg
2 1 1
cluster age sex chest_pain_type resting_blood_pressure cholestoral fasting_blood_sugar rest_ecg
1 1 59.05017 1.789298 3.608696 126.9130 228.8194 1.882943 2.387960
2 2 60.24473 1.540084 3.075949 150.8819 292.0633 1.751055 2.320675
3 3 48.79550 1.713701 2.950920 125.1452 234.1800 1.879346 2.625767
Max_heart_rate exercise_induced_angina oldpeak slope vessels_colored_by_flourosopy thalassemia target
1 125.6856 1.541806 1.9153846 1.923077 3.478261 2.795987 1.240803
2 148.1983 1.413502 1.3101266 1.679325 4.075949 2.552743 1.388186
3 163.8834 1.173824 0.4398773 1.394683 4.288344 1.940695 1.740286
1 2 3
299 237 489

8. Conclusion

The analysis of the heart disease dataset has provided significant insights into the various factors associated with heart disease risk. By employing different classification models such as logistic regression, decision tree, and random forest, we were able to predict heart disease with varying degrees of accuracy. The random forest model emerged as the most accurate, followed by logistic regression and decision tree models.Clustering analysis further revealed distinct groups within the data, providing a deeper understanding of how age, cholesterol levels, and other factors cluster among patients. The development and deployment of a user-friendly UI for heart disease analysis also enhanced the accessibility and usability of our findings.Our study underscores the importance of early diagnosis and accurate risk prediction in managing heart disease. By leveraging advanced data analytics techniques, we can contribute to better health outcomes and more targeted interventions for those at risk of heart disease.

9. Contribution

Student Name	Student ID	Contribution
SIMS Wimalasiri (Team leader)	29785	○ Clustering Models ○ Important Insights and Visualizations -Box plot ○ Heart Disease Analysis UI
SST Silva	30017	○ Introduction ○ Using libraries ○ Data preprocessing and cleaning - Assisted in building UI
KDT Kumara	29883	○ Important Insights and Visualizations - Prediction Decision Tree - Visualizations of Key Attributes - Assisted in building UI
RMDD Rathnayaka	29618	○ Classification Models -Model Accuracies -Confusion Matrices -Logistic Regression - Assisted in building UI
MVS Adikari	27613	○ Classification Models -Valuate the models on the test set -Random Forest -Decision Tree - Assisted in building UI

10. Reference

- www.kaggle.com. (n.d.). *Heart Disease Dataset UCI*. [online] Available at: https://www.kaggle.com/datasets/ketangangal/heart-disease-dataset-uci.

- rstudio.github.io. (n.d.). *Web Application Framework for R — shiny-package*. [online] Available at: https://rstudio.github.io/shiny/reference/shiny-package.html [Accessed 27 Jul. 2024].