Heart Disease Prediction Using Logistic Regression

Importing Packages :

```
library(dplyr)
library(tidyr)
library(readr)
library(caret)
library(ggplot2)
library(gplots)
library(GGally)
```

Importing Dataset

For this study, we will be using the heart disease dataset which includes patient data with a diagnosis of heart disease and is freely accessible on Kaggle. It consists of various factors pertaining to heart diseases such as age, sex, blood pressure, cholesterol level, blood sugar level, ECG results, etc. Based on the degree of blood vessel narrowing, the "target" variable, which represents our prediction target, is utilized to predict whether heart disease would manifest or not.

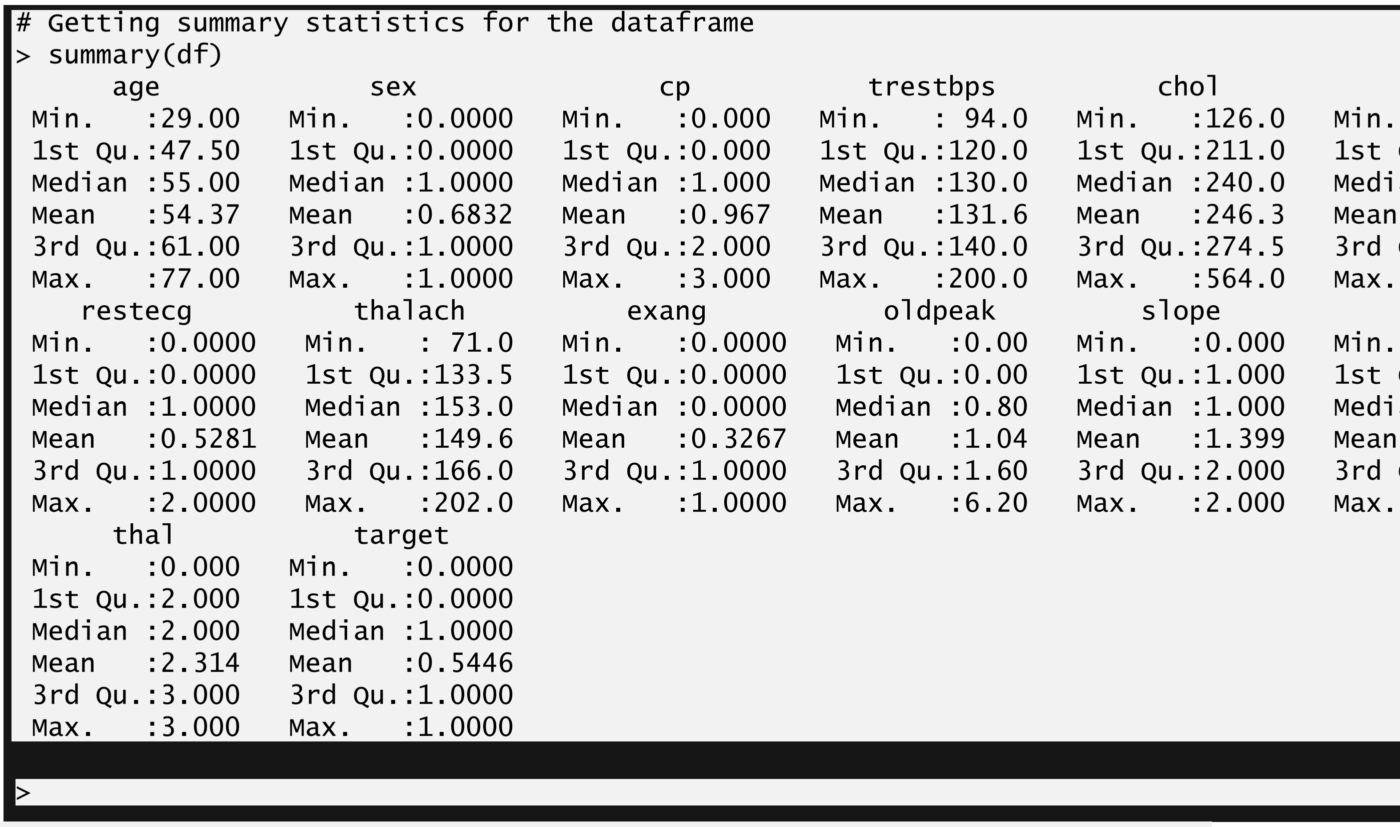
A target value of 0 indicates that the blood artery diameter is narrowing by less than 50%, implying a lower risk of heart disease.

A target value of 1 indicates that the narrowing is greater than 50%, implying an increased risk of heart disease.

```
df=read.csv("C:/Users/Top Prix/Downloads/dataset.csv")
> head(df)
  age sex cp trestbps chol fbs restecg thalach exang oldpeak slope ca thal target
1  63   1  3      145  233   1       0     150     0     2.3     0  0    1      1
2  37   1  2      130  250   0       1     187     0     3.5     0  0    2      1
3  41   0  1      130  204   0       0     172     0     1.4     2  0    2      1
4  56   1  1      120  236   0       1     178     0     0.8     2  0    2      1
5  57   0  0      120  354   0       1     163     1     0.6     2  0    2      1
6  57   1  0      140  192   0       1     148     0     0.4     1  0    1      1
```

```
# the dimensions of the dataframe
> dim(df)
[1] 303  14
```

From this we have learnt about the dimensions of our dataset. Our dataset has 303 rows of data available and for each row, we have 14 different features (vars).



```
# Getting summary statistics for the dataframe
> summary(df)
      age             sex               cp            trestbps         chol      
 Min.   :29.00   Min.   :0.0000   Min.   :0.000   Min.   : 94.0   Min.   :126.0   Min.
 1st Qu.:47.50   1st Qu.:0.0000   1st Qu.:0.000   1st Qu.:120.0   1st Qu.:211.0   1st 
 Median :55.00   Median :1.0000   Median :1.000   Median :130.0   Median :240.0   Medi
 Mean   :54.37   Mean   :0.6832   Mean   :0.967   Mean   :131.6   Mean   :246.3   Mean
 3rd Qu.:61.00   3rd Qu.:1.0000   3rd Qu.:2.000   3rd Qu.:140.0   3rd Qu.:274.5   3rd 
 Max.   :77.00   Max.   :1.0000   Max.   :3.000   Max.   :200.0   Max.   :564.0   Max.
    restecg          thalach          exang           oldpeak         slope      
 Min.   :0.0000   Min.   : 71.0   Min.   :0.0000   Min.   :0.00   Min.   :0.000   Min.
 1st Qu.:0.0000   1st Qu.:133.5   1st Qu.:0.0000   1st Qu.:0.00   1st Qu.:1.000   1st 
 Median :1.0000   Median :153.0   Median :0.0000   Median :0.80   Median :1.000   Medi
 Mean   :0.5281   Mean   :149.6   Mean   :0.3267   Mean   :1.04   Mean   :1.399   Mean
 3rd Qu.:1.0000   3rd Qu.:166.0   3rd Qu.:1.0000   3rd Qu.:1.60   3rd Qu.:2.000   3rd 
 Max.   :2.0000   Max.   :202.0   Max.   :1.0000   Max.   :6.20   Max.   :2.000   Max.
      thal           target      
 Min.   :0.000   Min.   :0.0000  
 1st Qu.:2.000   1st Qu.:0.0000  
 Median :2.000   Median :1.0000  
 Mean   :2.314   Mean   :0.5446  
 3rd Qu.:3.000   3rd Qu.:1.0000  
 Max.   :3.000   Max.   :1.0000  

>
```

```
> # Getting 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     : int  1 1 0 1 0 1 0 1 1 1 ...
 $ cp      : int  3 2 1 1 0 0 1 1 2 2 ...
 $ trestbps: 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 ...
 $ thalach : int  150 187 172 178 163 148 153 173 162 174 ...
 $ exang   : 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 ...
 $ slope   : int  0 0 2 2 2 1 1 2 2 2 ...
 $ ca      : int  0 0 0 0 0 0 0 0 0 0 ...
 $ thal    : int  1 2 2 2 2 1 2 3 3 2 ...
 $ target  : int  1 1 1 1 1 1 1 1 1 1 ...
```

We now have information about the type of each feature in our dataset.

Exploratory Data Analysis

EDA helps us in analyzing our data in detail and discover trends, patterns and other crucial information through statistical summaries and graphical representations. This understanding will help us in optimal training of the Logistic Regression model for better prediction accuracy.

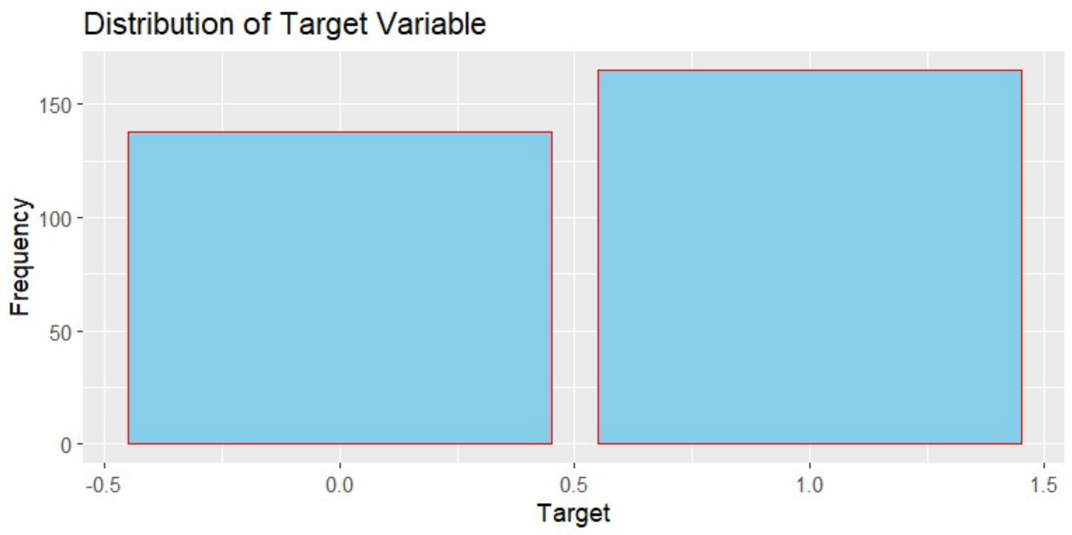
Firstly we will check if our dataset consists of any null values in any columns.

```
> # Count missing values in each column
> colSums(is.na(df))
      age       sex        cp  trestbps      chol       fbs   restecg   thalach     exang   oldp
        0         0         0         0         0         0         0         0         0
       ca      thal    target
        0         0         0

>
```

The output shows that there are no missing values in our heart disease dataset.

Now we will visualize our target variable using a barplot:

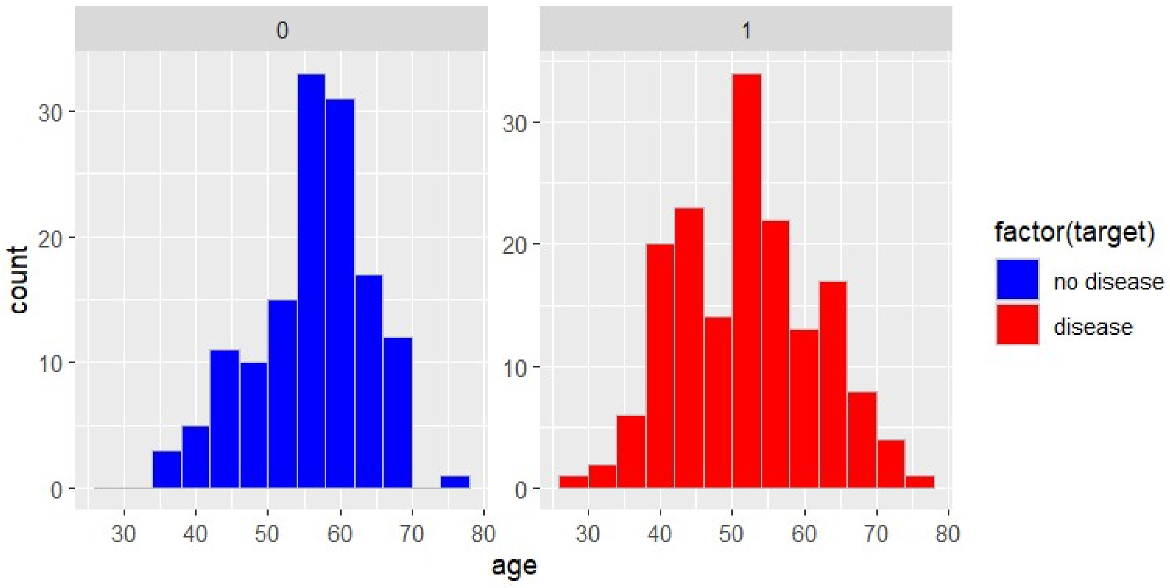
```
> # Visualizing the distribution of the target variable
> ggplot(df, aes(x = target)) +
+   geom_bar(fill = "skyblue", color = "red", stat = "count") +
+   labs(title = "Distribution of Target Variable", x = "Target", y = "Frequency")

>
```



We can observe from this plot that the number of instances of patients with heart disease is slightly more than those without heart disease.

Next we shall plot a histogram to show the relation between heart disease and age

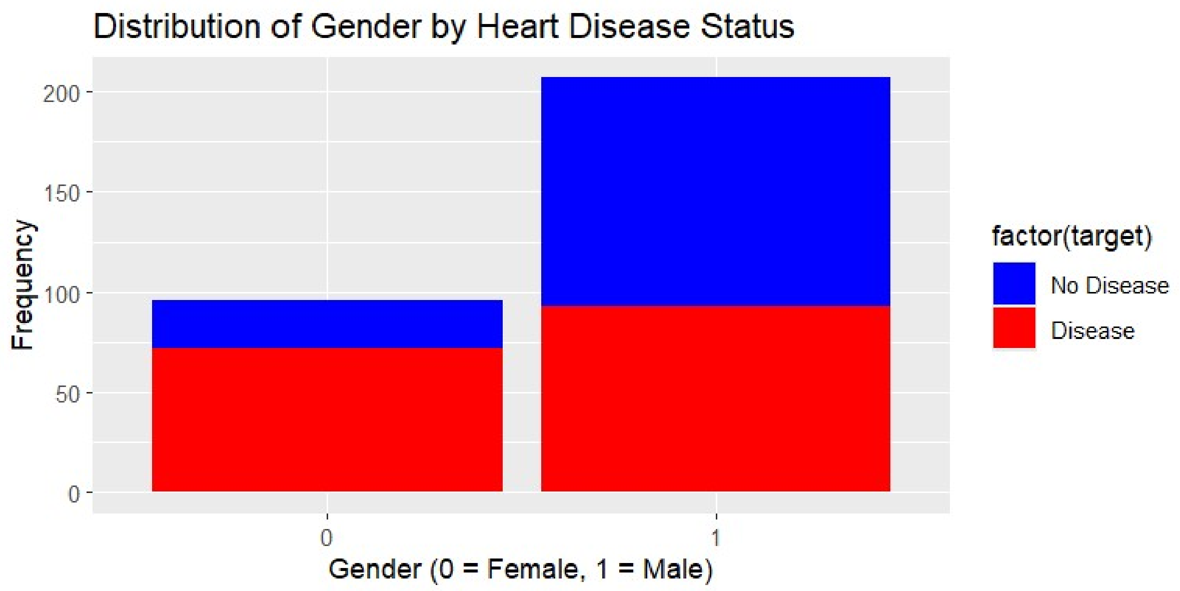
```
> # Creating two separate plots for heart disease and no heart disease
> ggplot(df,aes(x=age,fill=factor(target) ))+
+   geom_histogram(binwidth = 4, position = "dodge", color = 'grey')+
+   scale_fill_manual(values=c("0"="blue","1"="red"),labels=c("0"="no disease","1"="di
+   facet_wrap(~target, scales = "free_y")

>
```



We can observe that around 124 patients in the age of 50-60 had no heart diseases while around 100 patients in the age of 50-60 were suffering from heart diseases. This shows there is higher count of people without heart disease in ages of 50-60 in our dataset.

Next we will try to find a relation between gender and heart diseases

```
> # Visualizing the relationship between gender and heart disease
> ggplot(df, aes(x = factor(sex), fill = factor(target))) + geom_bar() +
+   labs(title = "Distribution of Gender by Heart Disease Status",
+        x = "Gender (0 = Female, 1 = Male)", y = "Frequency") +
+   scale_fill_manual(values = c("0" = "blue", "1" = "red"),
+                     labels = c("No Disease", "Disease"))

>
```

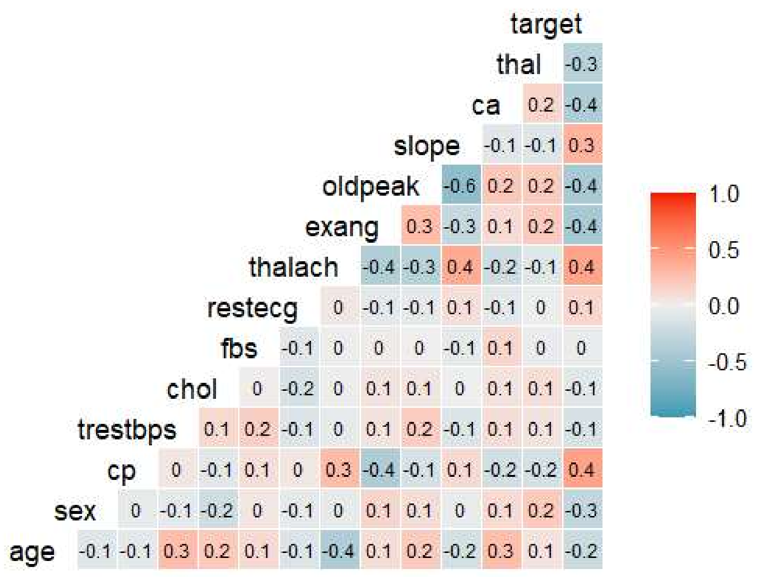


This visualization shows that the proportion of males with heart disease is much higher than females with heart disease showing a relation between gender and heart disease.

Correlation matrix Visualization

Finally, we use a correlation matrix to assess the relationships between the features. This matrix is a powerful tool for summarizing a large dataset, offering insights into patterns and potential dependencies among variables.

```
> # Correlation matrix
> ggcorr(df, label = TRUE, label_size = 2.5, hjust = 1, layout.exp = 2)

>
```



The variables "slope", "thalach" and "cp" have a positive correlation with the target variable so these must be having strong relation with heart disease. On the other hand, the variable "fbs" has 0 correlation indicating it doesn't have any relationship with our target variable.

Data Encoding

Exploratory Data Analysis reveals that certain factors like gender, type of chest pain, fasting blood sugar level, etc. are categorical variables and are not suitable for model training. We will encode these variables into "factor" data type in R for organized and improved understanding of this information. This encoding allows for efficient handling of categorical variables in statistical models and data analysis.

```
#factorisation de vars categorique
> heart <- df %>%
+   mutate(sex = as.factor(sex),
+          cp = as.factor(cp),
+          fbs = as.factor(fbs),
+          restecg = as.factor(restecg),
+          exang = as.factor(exang),
+          slope = as.factor(slope),
+          ca = as.factor(ca),
+          thal = as.factor(thal),
+          target = as.factor(target))
> #str
> str(heart)
'data.frame':	303 obs. of  14 variables:
 $ age     : int  63 37 41 56 57 57 56 44 52 57 ...
 $ sex     : Factor w/ 2 levels "0","1": 2 2 1 2 1 2 1 2 2 2 ...
 $ cp      : Factor w/ 4 levels "0","1","2","3": 4 3 2 2 1 1 2 2 3 3 ...
 $ trestbps: int  145 130 130 120 120 140 140 120 172 150 ...
 $ chol    : int  233 250 204 236 354 192 294 263 199 168 ...
 $ fbs     : Factor w/ 2 levels "0","1": 2 1 1 1 1 1 1 1 2 1 ...
 $ restecg : Factor w/ 3 levels "0","1","2": 1 2 1 2 2 2 1 2 2 2 ...
 $ thalach : int  150 187 172 178 163 148 153 173 162 174 ...
 $ exang   : Factor w/ 2 levels "0","1": 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 ...
 $ slope   : Factor w/ 3 levels "0","1","2": 1 1 3 3 3 2 2 3 3 3 ...
 $ ca      : Factor w/ 5 levels "0","1","2","3",..: 1 1 1 1 1 1 1 1 1 1 ...
 $ thal    : Factor w/ 4 levels "0","1","2","3": 2 3 3 3 3 2 3 4 4 3 ...
 $ target  : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 2 2 ...
```

Normalization and Data Splitting

Firstly we will select the features with higher relation with our target variable. Then we shall perform data normalization for stable and faster training of Logistic Regression model. After that, we split our dataset into two subsets. The initial subset, constituting 80% of the data, is employed to train the model, while the remaining 20% of the data is reserved for prediction purposes.

```
# Feature selection
> features <- df[, c('age', 'sex',  'cp', 'trestbps', 'chol', 'restecg', 'thalach',
+                    'exang', 'oldpeak', 'slope', 'ca', 'thal')]
> target <- df$target
> # Data normalization
> preprocessParams <- preProcess(features, method = c("center", "scale"))
> features_normalized <- predict(preprocessParams, features)
> # Splitting the data
> split <- createDataPartition(target, p = 0.8, list = FALSE)
> X_train <- features_normalized[split, ]
> X_test <- features_normalized[-split, ]
> Y_train <- target[split]
> Y_test <- target[-split]
>
> # Print the shape of the training and test sets
> print(paste("X_train shape:", paste(dim(X_train), collapse = "x")))
[1] "X_train shape: 243x12"
> print(paste("X_test shape:", paste(dim(X_test), collapse = "x")))
[1] "X_test shape: 60x12"
```

Now that we have our data normalized and split into train and test sets, we are ready to train the Logistic Regression model on this data.

```
# Combine features and target into a single data frame
> train_data <- as.data.frame(cbind(target = Y_train, X_train))
>
> # Training the logistic regression model
> model <- glm(target ~ ., data = train_data, family = "binomial")

>
```

Model Evaluation

After the model training is complete we can the predictions generated by Logistic Regression model on the test dataset. First we generate the probabilistic prediction by the Logistic Regression model and then we assign a threshold of 0.5 to further categorize these predictions into binary target classes.

```
# Making predictions on the test set
```

```
> predictions <- predict(model, newdata = as.data.frame(X_test), type = "r
esponse")
>
> # Converting probabilities to binary predictions based on threshold 0.5
> binary_predictions <- ifelse(predictions >= 0.5, 1, 0)
>
> # Combining actual values and predicted values into a data frame
> result <- data.frame(actual = Y_test, predicted = binary_predictions)
>
> # Evaluating the model
> confusionMatrix(data = as.factor(binary_predictions), reference = as.fac
tor(Y_test),
+                 positive = "1")
Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0 15  2
         1 13 30

               Accuracy : 0.75
                 95% CI : (0.6214, 0.8528)
    No Information Rate : 0.5333
    P-Value [Acc > NIR] : 0.0004647

                  Kappa : 0.4851

 Mcnemar's Test P-Value : 0.0098233

            Sensitivity : 0.9375
            Specificity : 0.5357
         Pos Pred Value : 0.6977
         Neg Pred Value : 0.8824
             Prevalence : 0.5333
         Detection Rate : 0.5000
   Detection Prevalence : 0.7167
      Balanced Accuracy : 0.7366

       'Positive' Class : 1
```

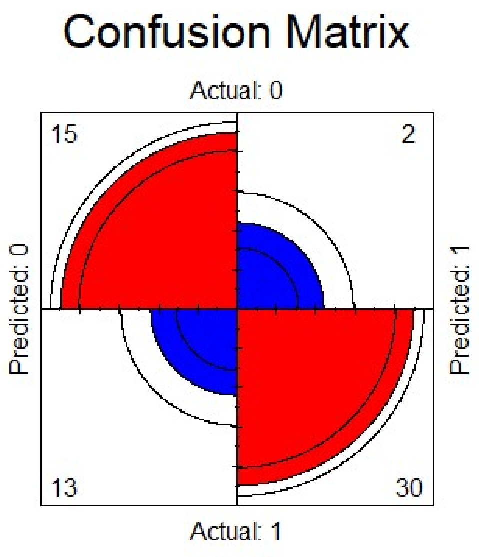
Here we have the model performance using evaluation metrics such as Accuracy, Confusion Matrix, Cohen's Kappa statistic, Sensitivity, etc. All these metrics help us in evaluating the performance of Logistic Regression on the heart disease dataset.

Here we can clearly see that our Logistic Regression model has achieved an accuracy of 75% on the given dataset.

Next we will also plot our confusion matrix to visualize the actual values against the predicted values.

```
# Create a confusion matrix
```

```
> conf_matrix <- table(factor(binary_predictions, levels = c("0", "1")),
+                      factor(Y_test, levels = c("0", "1")))
>
> # Set the dimension names of the confusion matrix
> dimnames(conf_matrix) <- list(Actual = c("0", "1"), Predicted = c("0", "1"))
>
> # Plot the fourfold plot with color and main title
> fourfoldplot(conf_matrix, color = c("blue", "red"), main = "Confusion Matrix")
```



Now We predict the values

```
# Combine features and target into a single data frame for the test set
> test_data <- as.data.frame(cbind(target = Y_test, X_test))
>
> # Making predictions on the test set
> predictions <- predict(model, newdata = as.data.frame(test_data[, -1]),t
ype ="response")
>
> # Converting probabilities to binary predictions based on threshold 0.5
> binary_predictions <- ifelse(predictions >= 0.5, 1, 0)
>
> # Combining actual values and predicted values into a data frame
> result <- data.frame(actual = test_data$target, predicted = binary_predi
ctions)
>
> # Displaying the results
> print(result)
   actual predicted
1       1         1
12      1         1
16      1         1
22      1         1
26      1         1
28      1         1
```

Conclusion

As a prevalent disease today, predicting heart disease in patients is crucial for timely intervention and recovery. Logistic regression offers a valuable method to predict heart disease by analyzing patterns in patient data. This information can assist healthcare professionals in identifying high-risk patients and implementing preventive measures.

Example:

```
# Example new patient
> new_patient <- data.frame(
+   age = 54,
+   sex = 1,
+   cp = 3,
+   trestbps = 130,
+   chol = 250,
+   restecg = 1,
+   thalach = 150,
+   exang = 0,
+   oldpeak = 1.5,
+   slope = 2,
+   ca = 0,
+   thal = 2
+ )
> # Normalize using same preProcess object used for training
> new_patient_norm <- predict(preprocessParams, new_patient)
> # Predict probability and class
> prob <- predict(model, newdata = new_patient_norm, type = "response")
> predicted_class <- ifelse(prob >= 0.5, 1, 0)
>
> # Output result
> cat("Predicted probability of heart disease:", round(prob, 3), "\n")
Predicted probability of heart disease: 0.965
> cat("Predicted class (0 = No disease, 1 = Disease):", predicted_class, "\n")
Predicted class (0 = No disease, 1 = Disease): 1
```