DATA SCIENCE PROGRAMMING

LAB (L3+L4)

ASSESSMENT 2

Logistic Regression IN R

Name: SWATHI D

Reg. no.: 22MID0035

1. Import Libraries and Dataset

```
#import libraries
library(dplyr)
library(tidyr)
library(readr)
library(caret)
library(ggplot2)
library(ggcorrplot)
library(tidyverse)
library(GGally)

#Load dataset
df <- read.csv("C:/Users/HP/Downloads/heart_disease_uci_3.csv",  header = TRUE, check.names = FALSE)
head(df)
```

```
> #import libraries
> library(dplyr)
> library(tidyr)
> library(readr)
> library(caret)
> library(ggplot2)
> library(ggcorrplot)
> library(tidyverse)
> library(GGally)
>
> #Load dataset
> df <- read.csv("C:/Users/HP/Downloads/heart_disease_uci_3.csv",  header = TRUE, check.names = FALSE)
> head(df)
  id age    sex   dataset             cp trestbps chol   fbs        restecg thalch exang oldpeak       slope
1  1  63   Male Cleveland  typical angina      145  233  TRUE lv hypertrophy    150 FALSE     2.3 downsloping
2  2  67   Male Cleveland    asymptomatic      160  286 FALSE lv hypertrophy    108  TRUE     1.5        flat
3  3  67   Male Cleveland    asymptomatic      120  229 FALSE lv hypertrophy    129  TRUE     2.6        flat
4  4  37   Male Cleveland     non-anginal      130  250 FALSE         normal    187 FALSE     3.5 downsloping
5  5  41 Female Cleveland atypical angina      130  204 FALSE lv hypertrophy    172 FALSE     1.4   upsloping
6  6  56   Male Cleveland atypical angina      120  236 FALSE         normal    178 FALSE     0.8   upsloping
  ca              thal target
1  0             fixed defect      0
2  3            normal      0
3  2 reversable defect      1
4  0            normal      0
5  0            normal      0
6  0            normal      0
```

2. Cleaning the dataset

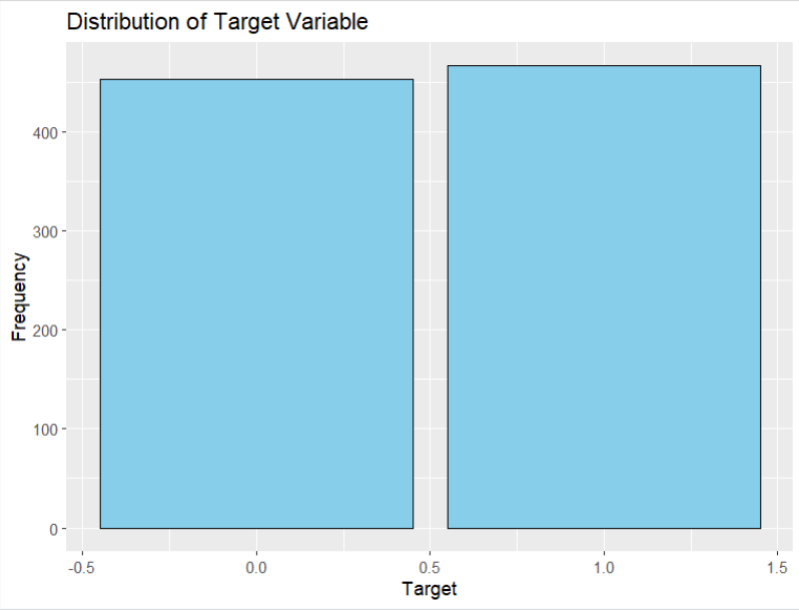
```r
#Cleaning the data
names = c("age", "sex", "cp", "trestbps", "chol", "fbs",
          "restecg", "thalach", "exang", "oldpeak", "slope", "ca", "thal", "heart_disease")
colnames(df) <- names
print(names)

names(df)
which(is.na(names(df)) | names(df) == "")

library(janitor)
df <- janitor::clean_names(df)

library(readr)
df <- read_csv("C:/Users/HP/Downloads/heart_disease_uci_3.csv", name_repair = "unique")

colSums(is.na(df))
str(df)
```

```
> #Cleaning the data
> names = c("age", "sex", "cp", "trestbps", "chol", "fbs",
+           "restecg", "thalach", "exang", "oldpeak", "slope", "ca", "thal", "heart_disease")
> colnames(df) <- names
> print(names)
 [1] "age"          "sex"          "cp"           "trestbps"     "chol"         "fbs"
 [7] "restecg"      "thalach"      "exang"        "oldpeak"      "slope"        "ca"
[13] "thal"         "heart_disease"
>
> names(df)
 [1] "age"          "sex"          "cp"           "trestbps"     "chol"         "fbs"
 [7] "restecg"      "thalach"      "exang"        "oldpeak"      "slope"        "ca"
[13] "thal"         "heart_disease" NA             NA
> which(is.na(names(df)) | names(df) == "")
[1] 15 16
>
> library(janitor)
> df <- janitor::clean_names(df)
>
> library(readr)
> df <- read_csv("C:/Users/HP/Downloads/heart_disease_uci_3.csv", name_repair = "unique")
Rows: 920 Columns: 16
── Column specification ────────────────────────────────────────────────────────
Delimiter: ","
chr (6): sex, dataset, cp, restecg, slope, thal
dbl (8): id, age, trestbps, chol, thalch, oldpeak, ca, target
lgl (2): fbs, exang

ℹ Use `spec()` to retrieve the full column specification for this data.
ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
>
> colSums(is.na(df))
      id      age      sex  dataset       cp trestbps     chol      fbs  restecg   thalch    exang  oldpeak
       0        0        0        0        0       59       30       90        2       55       55       62
   slope       ca     thal   target
     309      611      486        0
```

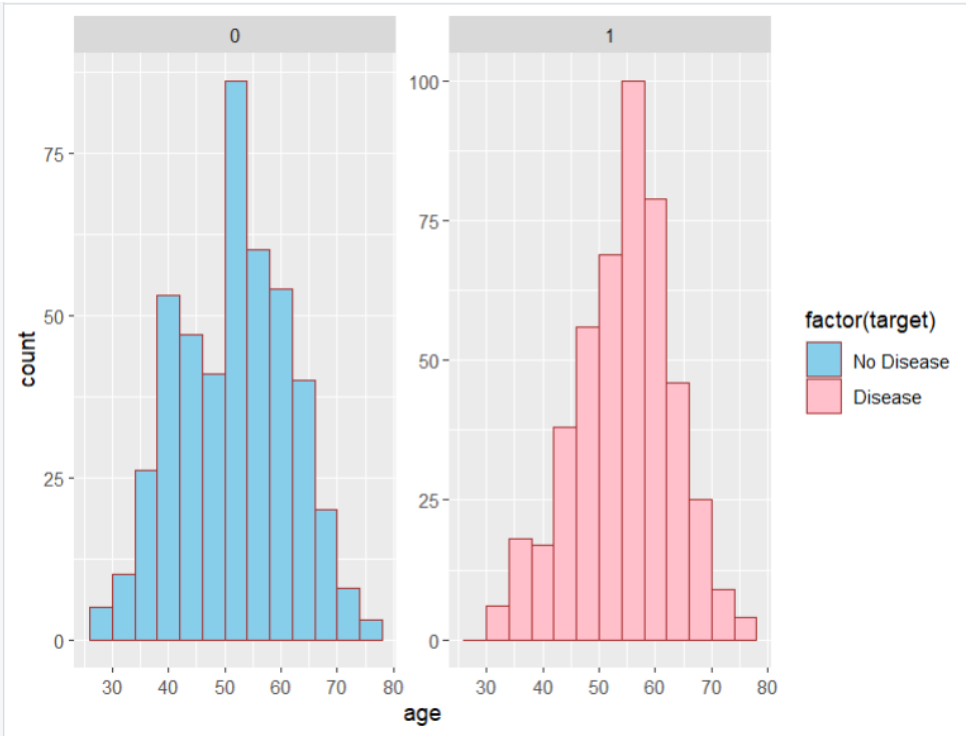
```
> str(df)
spc_tbl_ [920 × 16] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
 $ id      : num [1:920] 1 2 3 4 5 6 7 8 9 10 ...
 $ age     : num [1:920] 63 67 67 37 41 56 62 57 63 53 ...
 $ sex     : chr [1:920] "Male" "Male" "Male" "Male" ...
 $ dataset : chr [1:920] "Cleveland" "Cleveland" "Cleveland" "Cleveland" ...
 $ cp      : chr [1:920] "typical angina" "asymptomatic" "asymptomatic" "non-anginal" ...
 $ trestbps: num [1:920] 145 160 120 130 130 120 140 120 130 140 ...
 $ chol    : num [1:920] 233 286 229 250 204 236 268 354 254 203 ...
 $ fbs     : logi [1:920] TRUE FALSE FALSE FALSE FALSE FALSE ...
 $ restecg : chr [1:920] "lv hypertrophy" "lv hypertrophy" "lv hypertrophy" "normal" ...
 $ thalch  : num [1:920] 150 108 129 187 172 178 160 163 147 155 ...
 $ exang   : logi [1:920] FALSE TRUE TRUE FALSE FALSE FALSE ...
 $ oldpeak : num [1:920] 2.3 1.5 2.6 3.5 1.4 0.8 3.6 0.6 1.4 3.1 ...
 $ slope   : chr [1:920] "downsloping" "flat" "flat" "downsloping" ...
 $ ca      : num [1:920] 0 3 2 0 0 0 2 0 1 0 ...
 $ thal    : chr [1:920] "fixed defect" "normal" "reversable defect" "normal" ...
 $ target  : num [1:920] 0 0 1 0 0 0 1 0 1 1 ...
 - attr(*, "spec")=
  .. cols(
  ..   id = col_double(),
  ..   age = col_double(),
  ..   sex = col_character(),
  ..   dataset = col_character(),
  ..   cp = col_character(),
  ..   trestbps = col_double(),
  ..   chol = col_double(),
  ..   fbs = col_logical(),
  ..   restecg = col_character(),
  ..   thalch = col_double(),
  ..   exang = col_logical(),
  ..   oldpeak = col_double(),
  ..   slope = col_character(),
  ..   ca = col_double(),
  ..   thal = col_character(),
  ..   target = col_double()
  .. )
 - attr(*, "problems")=<externalptr>
```

3. Visualizations

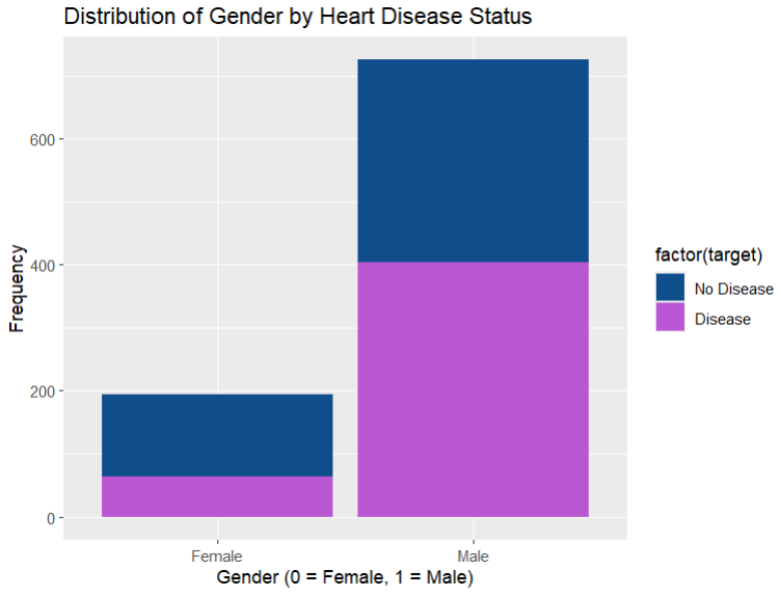
```
# Visualizing the distribution of the target variable
ggplot(df, aes(x = target)) +
  geom_bar(fill = "skyblue", color = "black", stat = "count") +
  labs(title = "Distribution of Target Variable", x = "Target", y = "Frequency")
```



```r
# 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 = 'brown') +
  scale_fill_manual(values = c("0" = "skyblue", "1" = "pink"),
                    labels = c("No Disease", "Disease")) +
  facet_wrap(~target, scales = "free_y")
```

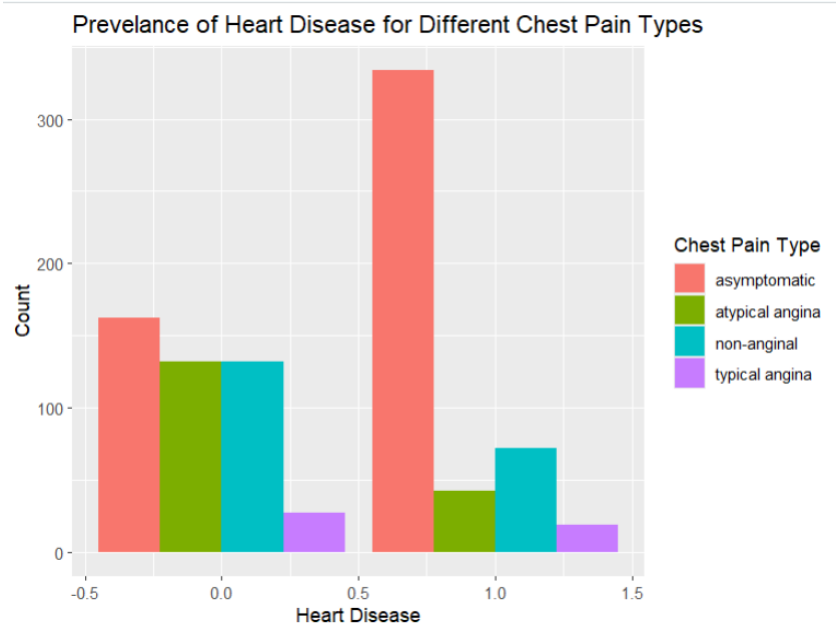



```r
# Visualizing the relationship between gender and heart disease
ggplot(df, aes(x = factor(sex), fill = factor(target))) + geom_bar() +
  labs(title = "Distribution of Gender by Heart Disease Status",
       x = "Gender (0 = Female, 1 = Male)", y = "Frequency") +
  scale_fill_manual(values = c("0" = "dodgerblue4", "1" = "mediumorchid"),
                    labels = c("No Disease", "Disease"))
```



```r
# Correlation matrix
numeric_df <- df %>% select(where(is.numeric))

# Plot correlation matrix with custom colors
ggcorr(numeric_df,
       label = TRUE,
       label_size = 3,
       hjust = 0.75,
       layout.exp = 1,
       low = "brown",
       mid = "skyblue",
       high = "pink")
```


```r
cp.plot <- ggplot(df, mapping = aes(x=target, fill = cp)) +
  geom_bar(position = "dodge") +
  labs(title = "Prevelance of Heart Disease for Different Chest
       Pain Types", x = "Heart Disease", y = "Count", fill = "Chest Pain Type")

cp.plot
```



4. Encoder

```r
# encoding
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))

# Checking the structure of the data set
str(heart)
```

```
> 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))
>
> # Checking the structure of the data set
> str(heart)
tibble [920 × 16] (S3: tbl_df/tbl/data.frame)
 $ id      : num [1:920] 1 2 3 4 5 6 7 8 9 10 ...
 $ age     : num [1:920] 63 67 67 37 41 56 62 57 63 53 ...
 $ sex     : Factor w/ 2 levels "Female","Male": 2 2 2 2 1 2 1 1 2 2 ...
 $ dataset : chr [1:920] "Cleveland" "Cleveland" "Cleveland" "Cleveland" ...
 $ cp      : Factor w/ 4 levels "asymptomatic",..: 4 1 1 3 2 2 1 1 1 1 ...
 $ trestbps: num [1:920] 145 160 120 130 130 120 140 120 130 140 ...
 $ chol    : num [1:920] 233 286 229 250 204 236 268 354 254 203 ...
 $ fbs     : Factor w/ 2 levels "FALSE","TRUE": 2 1 1 1 1 1 1 1 1 2 ...
 $ restecg : Factor w/ 3 levels "lv hypertrophy",..: 1 1 1 2 1 2 1 2 1 1 ...
 $ thalch  : num [1:920] 150 108 129 187 172 178 160 163 147 155 ...
 $ exang   : Factor w/ 2 levels "FALSE","TRUE": 1 2 2 1 1 1 1 2 1 2 ...
 $ oldpeak : num [1:920] 2.3 1.5 2.6 3.5 1.4 0.8 3.6 0.6 1.4 3.1 ...
 $ slope   : Factor w/ 3 levels "downsloping",..: 1 2 2 1 3 3 1 3 2 1 ...
 $ ca      : Factor w/ 4 levels "0","1","2","3": 1 4 3 1 1 1 3 1 2 1 ...
 $ thal    : Factor w/ 3 levels "fixed defect",..: 1 2 3 2 2 2 2 2 3 3 ...
 $ target  : Factor w/ 2 levels "0","1": 1 1 2 1 1 1 2 1 2 2 ...
```

5. Features scaling and normalization

```r
# Feature selection
features <- df[, c('age', 'sex',  'cp', 'trestbps', 'chol', 'restecg', 'thal',
                   'exang', 'oldpeak', 'slope', 'ca', 'thal')]
target <- df$target


# Data normalization
preprocessParams <- preProcess(features, method = c("center", "scale"))
features_normalized <- predict(preprocessParams, features)
```

```
> # Feature selection
> features <- df[, c('age', 'sex',  'cp', 'trestbps', 'chol', 'restecg', 'thal',
+                    'exang', 'oldpeak', 'slope', 'ca', 'thal')]
> target <- df$target
>
>
> # Data normalization
> preprocessParams <- preProcess(features, method = c("center", "scale"))
> features_normalized <- predict(preprocessParams, features)
> # Feature selection
> features <- df[, c('age', 'sex',  'cp', 'trestbps', 'chol', 'restecg', 'thal',
+                    'exang', 'oldpeak', 'slope', 'ca', 'thal')]
> target <- df$target
>
>
> # Data normalization
> preprocessParams <- preProcess(features, method = c("center", "scale"))
> features_normalized <- predict(preprocessParams, features)
>
```

6. Training and testing

```
#splitting the data
set.seed(123)
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(paste("X_train shape:", paste(dim(X_train), collapse = "x")))
print(paste("X_test shape:", paste(dim(X_test), collapse = "x")))

#build train test
train_data <- data.frame(X_train, target = Y_train)
test_data  <- data.frame(X_test,  target = Y_test)

train_data <- train_data[, !duplicated(names(train_data))]
test_data  <- test_data[, !duplicated(names(test_data))]
str(heart)

any(duplicated(names(train_data)))
names(train_data)[duplicated(names(train_data))]
```

```
> #splitting the data
> set.seed(123)
> 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(paste("X_train shape:", paste(dim(X_train), collapse = "x")))
[1] "X_train shape: 736x12"
> print(paste("X_test shape:", paste(dim(X_test), collapse = "x")))
[1] "X_test shape: 184x12"
>
> #build train test
> train_data <- data.frame(X_train, target = Y_train)
> test_data  <- data.frame(X_test,  target = Y_test)
>
> train_data <- train_data[, !duplicated(names(train_data))]
> test_data  <- test_data[, !duplicated(names(test_data))]
```

```
> str(heart)
tibble [920 × 16] (S3: tbl_df/tbl/data.frame)
 $ id      : num [1:920] 1 2 3 4 5 6 7 8 9 10 ...
 $ age     : num [1:920] 63 67 67 37 41 56 62 57 63 53 ...
 $ sex     : Factor w/ 2 levels "Female","Male": 2 2 2 2 1 2 1 1 2 2 ...
 $ dataset : chr [1:920] "Cleveland" "Cleveland" "Cleveland" "Cleveland" ...
 $ cp      : Factor w/ 4 levels "asymptomatic",..: 4 1 1 3 2 2 1 1 1 1 ...
 $ trestbps: num [1:920] 145 160 120 130 130 120 140 120 130 140 ...
 $ chol    : num [1:920] 233 286 229 250 204 236 268 354 254 203 ...
 $ fbs     : Factor w/ 2 levels "FALSE","TRUE": 2 1 1 1 1 1 1 1 1 2 ...
 $ restecg : Factor w/ 3 levels "lv hypertrophy",..: 1 1 1 2 1 2 1 2 1 1 ...
 $ thalch  : num [1:920] 150 108 129 187 172 178 160 163 147 155 ...
 $ exang   : Factor w/ 2 levels "FALSE","TRUE": 1 2 2 1 1 1 1 2 1 2 ...
 $ oldpeak : num [1:920] 2.3 1.5 2.6 3.5 1.4 0.8 3.6 0.6 1.4 3.1 ...
 $ slope   : Factor w/ 3 levels "downsloping",..: 1 2 2 1 3 3 1 3 2 1 ...
 $ ca      : Factor w/ 4 levels "0","1","2","3": 1 4 3 1 1 1 3 1 2 1 ...
 $ thal    : Factor w/ 3 levels "fixed defect",..: 1 2 3 2 2 2 2 2 3 3 ...
 $ target  : Factor w/ 2 levels "0","1": 1 1 2 1 1 1 2 1 2 2 ...
>
> any(duplicated(names(train_data)))
[1] FALSE
> names(train_data)[duplicated(names(train_data))]
character(0)
```

```r
# Align factor levels (important!)
for (col in names(train_data)) {
  if (is.factor(train_data[[col]])) {
    test_data[[col]] <- factor(test_data[[col]], levels = levels(train_data[[col]]))
  }
}

# Print the shape of the training and test sets
print(paste("X_train shape:", paste(dim(X_train), collapse = "x")))
print(paste("X_test shape:", paste(dim(X_test), collapse = "x")))
```

```
> # Align factor levels (important!)
> for (col in names(train_data)) {
+   if (is.factor(train_data[[col]])) {
+     test_data[[col]] <- factor(test_data[[col]], levels = levels(train_data[[col]]))
+   }
+ }
>
> # Print the shape of the training and test sets
> print(paste("X_train shape:", paste(dim(X_train), collapse = "x")))
[1] "X_train shape: 736x12"
> print(paste("X_test shape:", paste(dim(X_test), collapse = "x")))
[1] "X_test shape: 184x12"
```

7. Implement the model

```r
#LOGISTIC REGRESSION MODEL
# Combine features and target into a single data frame
train_data <- as.data.frame(cbind(target = Y_train, X_train))
train_data <- train_data[, !duplicated(names(train_data))]

# Fit logistic regression model
model <- glm(target ~ ., data = train_data, family = "binomial")
summary(model)
```

```
> #LOGISTIC REGRESSION MODEL
> # Combine features and target into a single data frame
> train_data <- as.data.frame(cbind(target = Y_train, X_train))
> train_data <- train_data[, !duplicated(names(train_data))]
>
> # Fit logistic regression model
> model <- glm(target ~ ., data = train_data, family = "binomial")
```

```
> summary(model)

Call:
glm(formula = target ~ ., family = "binomial", data = train_data)

Coefficients:
                          Estimate Std. Error z value Pr(>|z|)
(Intercept)               0.067797   1.233127   0.055 0.956154
age                       0.005288   0.221636   0.024 0.980964
sexMale                   0.746201   0.516027   1.446 0.148163
cpatypical angina        -0.558182   0.553154  -1.009 0.312931
cpnon-anginal            -1.734670   0.501198  -3.461 0.000538 ***
cptypical angina         -0.928437   0.642218  -1.446 0.148269
trestbps                  0.229049   0.212653   1.077 0.281436
chol                      0.138762   0.417517   0.332 0.739624
restecgnormal            -0.560020   0.385438  -1.453 0.146239
restecgst-t abnormality   0.717995   1.956182   0.367 0.713590
thalnormal               -0.583122   0.726874  -0.802 0.422419
thalreversable defect     0.778001   0.704302   1.105 0.269315
exangTRUE                 0.716526   0.429034   1.670 0.094901 .
oldpeak                   0.291409   0.242088   1.204 0.228693
slopeflat                 0.348099   0.824438   0.422 0.672860
slopeupsloping           -0.900451   0.876639  -1.027 0.304344
ca                        0.847160   0.232850   3.638 0.000275 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 335.71  on 244  degrees of freedom
Residual deviance: 190.61  on 228  degrees of freedom
  (491 observations deleted due to missingness)
AIC: 224.61

Number of Fisher Scoring iterations: 5
```

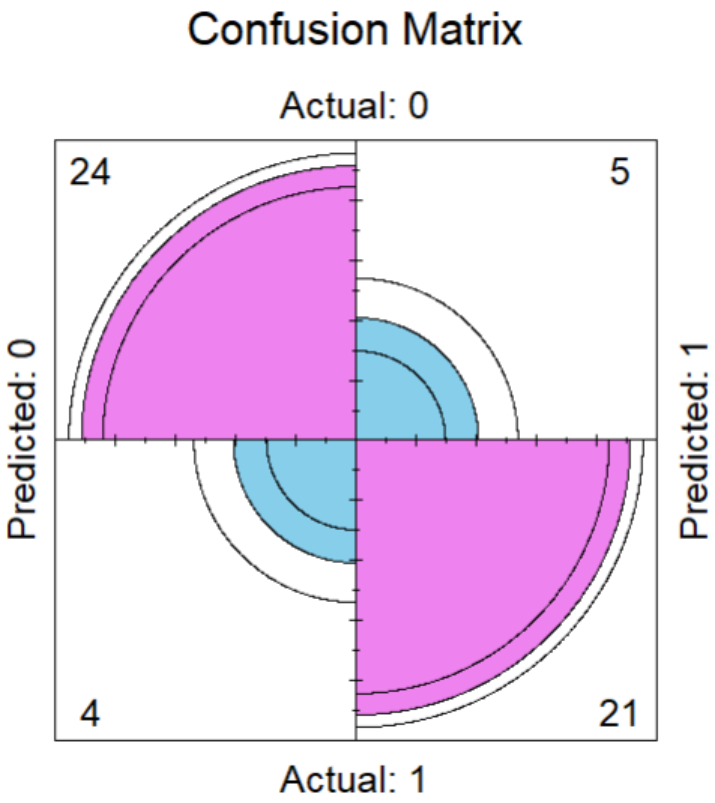
8. Model evaluation

```
#model evaluation
# Making predictions on the test set
predictions <- predict(model, newdata =test_data, type = "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 = Y_test, predicted = binary_predictions)

# Evaluating the model
confusionMatrix(data = as.factor(binary_predictions), reference = as.factor(Y_test),
                positive = "1")
```

```
> #model evaluation
> # Making predictions on the test set
> predictions <- predict(model, newdata =test_data, type = "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 = Y_test, predicted = binary_predictions)
>
> # Evaluating the model
> confusionMatrix(data = as.factor(binary_predictions), reference = as.factor(Y_test),
+                 positive = "1")
Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0 24  5
         1  4 21

               Accuracy : 0.8333
                 95% CI : (0.7071, 0.9208)
    No Information Rate : 0.5185
    P-Value [Acc > NIR] : 1.357e-06

                  Kappa : 0.6657

 Mcnemar's Test P-Value : 1

            Sensitivity : 0.8077
            Specificity : 0.8571
         Pos Pred Value : 0.8400
         Neg Pred Value : 0.8276
             Prevalence : 0.4815
         Detection Rate : 0.3889
   Detection Prevalence : 0.4630
      Balanced Accuracy : 0.8324

       'Positive' Class : 1
```

9. Confusion matrix

```r
# Create a confusion matrix
conf_matrix <- table(
  factor(binary_predictions, levels = c("0", "1")),
  factor(test_data$target, 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("skyblue", "violet"), main = "Confusion Matrix")
```



10. Predict values

```
#predict new values for the model
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]),type ="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_predictions)

# Displaying the results
print(result)
```

```
> # Making predictions on the test set
> predictions <- predict(model, newdata = as.data.frame(test_data[, -1]),type ="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_predictions)
```

```
> # Displaying the results
> print(result)
    actual predicted
1        0         0
2        1         1
3        1         1
4        1         1
5        0         0
6        0         0
7        0         1
8        0         0
9        0         0
10       1         1
11       0         0
12       0         0
13       0         0
14       0         0
15       1         1
16       1         0
17       0         0
18       0         0
19       1         1
20       0         0
21       1         1
22       1         1
23       0         0
24       0         0
25       0         0
26       1         1
27       0         0
28       1         0
29       1         1
30       0         0
31       0         0
32       1         1
33       1         1
34       1         1
35       1         1
36       0         0
37       0         1
38       1         1
39       0         0
40       1         1
41       0         0
42       1         1
43       1         0
44       0         1
45       0         0
46       1         0
47       1         1
48       0         1
49       0         0
50       0         0
51       1         0
52       1         1
53       1         1
54       0        NA
55       0        NA
56       0        NA
57       0        NA
58       0        NA
59       0        NA
60       0        NA
61       0        NA
62       0        NA
63       0        NA
64       0        NA
65       0        NA
66       0        NA
67       0        NA
68       0        NA
69       0        NA
70       1        NA
71       0        NA
72       1        NA
73       0        NA
74       1        NA
75       0        NA
76       0        NA
77       0        NA
78       0        NA
79       0        NA
80       0        NA
81       0        NA
82       1        NA
83       1        NA
84       1        NA
85       0        NA
86       1        NA
87       0        NA
88       1        NA
89       1        NA
90       1        NA
91       1        NA
92       1        NA
93       1        NA
94       1        NA
95       1        NA
96       1        NA
97       1        NA
98       1        NA
99       1        NA
100      1        NA
101      1        NA
102      1        NA
103      1        NA
104      1        NA
105      1        NA
106      1        NA
107      1        NA
108      1        NA
109      1        NA
110      0        NA
111      1        NA
112      1        NA
113      1        NA
114      0        NA
115      1        NA
116      0        NA
117      0        NA
118      1        NA
119      1        NA
120      0        NA
121      0        NA
122      1        NA
123      1        NA
124      1        NA
125      0        NA
126      1        NA
127      0        NA
128      0        NA
129      0        NA
130      1        NA
131      0        NA
132      0        NA
133      1        NA
134      0        NA
135      0        NA
136      1        NA
137      0        NA
138      1        NA
139      1        NA
140      0        NA
141      0        NA
142      0        NA
143      0        NA
144      1        NA
145      0        NA
146      1        NA
147      1        NA
148      0        NA
149      0        NA
150      1        NA
151      1        NA
152      1        NA
153      1         1
154      1        NA
155      1        NA
156      0        NA
157      1        NA
158      0        NA
159      0        NA
160      1        NA
161      1        NA
162      1        NA
163      1        NA
164      1        NA
165      1        NA
166      0        NA
167      1        NA
168      1        NA
169      1        NA
170      0        NA
171      0        NA
172      1        NA
173      1        NA
174      1        NA
175      1        NA
176      1        NA
177      0        NA
178      1        NA
179      1        NA
180      0        NA
181      0        NA
182      1        NA
183      0        NA
184      1        NA
```