HEART DISEASE PREDICTIONS (LOGISTIC REGRESSION MODEL)

Ajinkya

2024-12-24

```
# Load the tidyverse package, which includes useful packages for data manipulation and visualization

library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 4.3.3
```

```
## Warning: package 'ggplot2' was built under R version 4.3.3
```

```
## Warning: package 'tidyr' was built under R version 4.3.3
```

```
## Warning: package 'dplyr' was built under R version 4.4.0
```

```
## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
## ✓ dplyr     1.1.4     ✓ readr     2.1.5
## ✓ forcats   1.0.0     ✓ stringr   1.5.1
## ✓ ggplot2   3.5.1     ✓ tibble    3.2.1
## ✓ lubridate 1.9.3     ✓ tidyr     1.3.1
## ✓ purrr     1.0.2
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✗ dplyr::filter() masks stats::filter()
## ✗ dplyr::lag()    masks stats::lag()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

Step 1: Read the heart.csv file from the specified file path and store the data in the ‘heart’ data frame

```
heart <- read_csv("C:/Users/Ajinkyaa/OneDrive/Stata to R/New folder/HEART DISEASE PREDICTION MODEL/heart.csv")
```

```
## Rows: 1025 Columns: 14
## ── Column specification ────────────────────────────────────────────────────────
## Delimiter: ","
## dbl (14): age, sex, cp, trestbps, chol, fbs, restecg, thalach, exang, oldpea...
##
## 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.
```

Step 2: Data Exploration, to Understand data structure, identify problems (missing/duplicate values), and ensure data integrity

```
view(heart) #Browse the all loaded data
head(heart) #Viewing the First Few Rows
```

```
## # A tibble: 6 × 14
##     age   sex    cp trestbps  chol   fbs restecg thalach exang oldpeak slope
##   <dbl> <dbl> <dbl>    <dbl> <dbl> <dbl>   <dbl>   <dbl> <dbl>   <dbl> <dbl>
## 1    52     1     0      125   212     0       1     168     0     1       2
## 2    53     1     0      140   203     1       0     155     1     3.1     0
## 3    70     1     0      145   174     0       1     125     1     2.6     0
## 4    61     1     0      148   203     0       1     161     0     0       2
## 5    62     0     0      138   294     1       1     106     0     1.9     1
## 6    58     0     0      100   248     0       0     122     0     1       1
## # ℹ 3 more variables: ca <dbl>, thal <dbl>, target <dbl>
```

```
glimpse(heart) #Summarizing Data Structure (Provides an overview of the dataset, including the number of rows and columns, d
ata types (numeric, categorical, etc.), and a sample of the data in each column)
```

```
## Rows: 1,025
## Columns: 14
## $ age      <dbl> 52, 53, 70, 61, 62, 58, 58, 55, 46, 54, 71, 43, 34, 51, 52, 3…
## $ sex      <dbl> 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 1…
## $ cp       <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 2, 0, 1, 2, 2…
## $ trestbps <dbl> 125, 140, 145, 148, 138, 100, 114, 160, 120, 122, 112, 132, 1…
## $ chol     <dbl> 212, 203, 174, 203, 294, 248, 318, 289, 249, 286, 149, 341, 2…
## $ fbs      <dbl> 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0…
## $ restecg  <dbl> 1, 0, 1, 1, 1, 0, 2, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0…
## $ thalach  <dbl> 168, 155, 125, 161, 106, 122, 140, 145, 144, 116, 125, 136, 1…
## $ exang    <dbl> 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0…
## $ oldpeak  <dbl> 1.0, 3.1, 2.6, 0.0, 1.9, 1.0, 4.4, 0.8, 0.8, 3.2, 1.6, 3.0, 0…
## $ slope    <dbl> 2, 0, 0, 2, 1, 1, 0, 1, 2, 1, 1, 1, 2, 1, 1, 2, 2, 1, 2, 2, 1…
## $ ca       <dbl> 2, 0, 0, 1, 3, 0, 3, 1, 0, 2, 0, 0, 0, 3, 0, 0, 1, 1, 0, 0, 0…
## $ thal     <dbl> 3, 3, 3, 3, 2, 2, 1, 3, 3, 2, 2, 3, 2, 3, 0, 2, 2, 3, 2, 2, 2…
## $ target   <dbl> 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0…
```

```
anyNA(heart) #Checking for Missing Values
```

```
## [1] FALSE
```

```
duplicated(heart) %>% table() # Identifying Duplicate Rows (Counts rows that are exact duplicates)
```

```
## .
## FALSE  TRUE
##   302   723
```

```
heart <- heart %>% distinct() #Removing Duplicates (Keeps only unique rows, removing duplicates)
```

Step 3: Data Relationships, To calculate the correlation matrix between numerical variables in the 'heart' data frame and round it to 2 decimal places

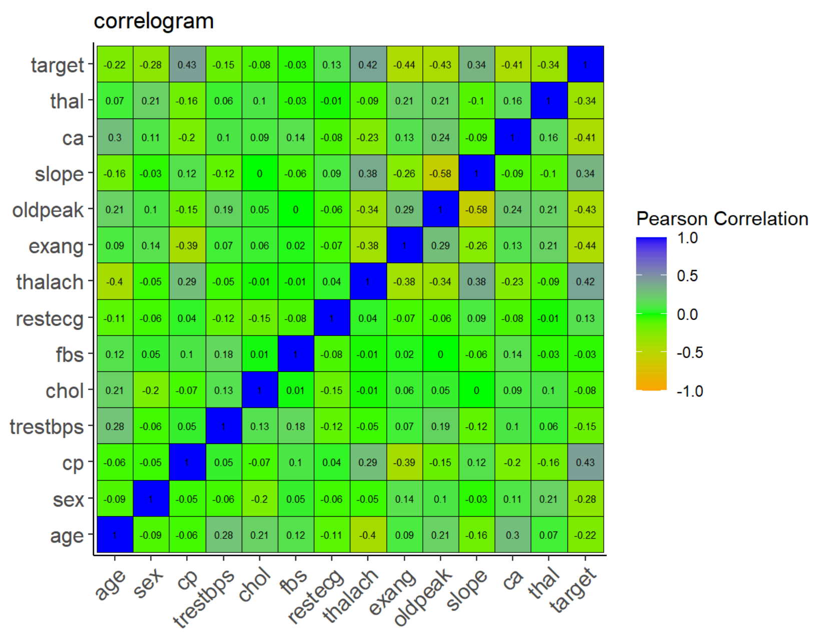
```
round(cor(heart), 2) #Checking Relationships with Correlation
```

```
##            age   sex    cp trestbps  chol   fbs restecg thalach exang oldpeak
## age       1.00 -0.09 -0.06     0.28  0.21  0.12   -0.11   -0.40  0.09    0.21
## sex      -0.09  1.00 -0.05    -0.06 -0.20  0.05   -0.06   -0.05  0.14    0.10
## cp       -0.06 -0.05  1.00     0.05 -0.07  0.10    0.04    0.29 -0.39   -0.15
## trestbps  0.28 -0.06  0.05     1.00  0.13  0.18   -0.12   -0.05  0.07    0.19
## chol      0.21 -0.20 -0.07     0.13  1.00  0.01   -0.15   -0.01  0.06    0.05
## fbs       0.12  0.05  0.10     0.18  0.01  1.00   -0.08   -0.01  0.02    0.00
## restecg  -0.11 -0.06  0.04    -0.12 -0.15 -0.08    1.00    0.04 -0.07   -0.06
## thalach  -0.40 -0.05  0.29    -0.05 -0.01 -0.01    0.04    1.00 -0.38   -0.34
## exang     0.09  0.14 -0.39     0.07  0.06  0.02   -0.07   -0.38  1.00    0.29
## oldpeak   0.21  0.10 -0.15     0.19  0.05  0.00   -0.06   -0.34  0.29    1.00
## slope    -0.16 -0.03  0.12    -0.12  0.00 -0.06    0.09    0.38 -0.26   -0.58
## ca        0.30  0.11 -0.20     0.10  0.09  0.14   -0.08   -0.23  0.13    0.24
## thal      0.07  0.21 -0.16     0.06  0.10 -0.03   -0.01   -0.09  0.21    0.21
## target   -0.22 -0.28  0.43    -0.15 -0.08 -0.03    0.13    0.42 -0.44   -0.43
##          slope    ca  thal target
## age      -0.16  0.30  0.07  -0.22
## sex      -0.03  0.11  0.21  -0.28
## cp        0.12 -0.20 -0.16   0.43
## trestbps -0.12  0.10  0.06  -0.15
## chol      0.00  0.09  0.10  -0.08
## fbs      -0.06  0.14 -0.03  -0.03
## restecg   0.09 -0.08 -0.01   0.13
## thalach   0.38 -0.23 -0.09   0.42
## exang    -0.26  0.13  0.21  -0.44
## oldpeak  -0.58  0.24  0.21  -0.43
## slope     1.00 -0.09 -0.10   0.34
## ca       -0.09  1.00  0.16  -0.41
## thal     -0.10  0.16  1.00  -0.34
## target    0.34 -0.41 -0.34   1.00
```

```
cr <- round(cor(heart), 2) #Store correlation matrix

#Visualize your correlations (Creates a visual heatmap of the correlation matrix)
#install.packages("ggcorrplot")
library(ggcorrplot)
```

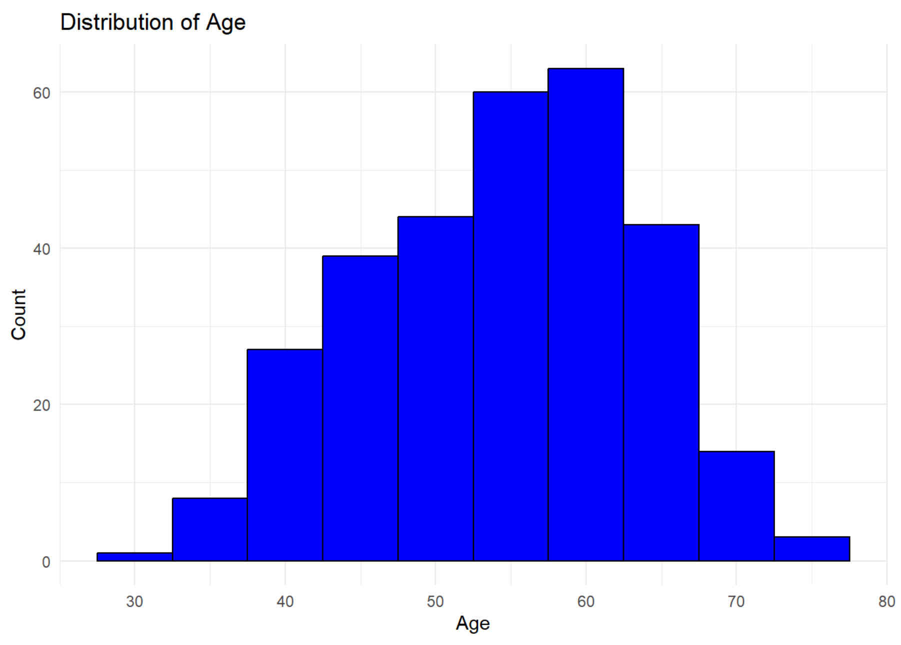
```
## Warning: package 'ggcorrplot' was built under R version 4.3.3
```

```
ggcorrplot(cr,title = "correlogram",lab_col = "black",
           lab = TRUE, legend.title = "Pearson Correlation",
           lab_size = 2, ggtheme = theme_classic(),
           outline.color = "black",
           colors = c("orange", "green", "blue"))
```

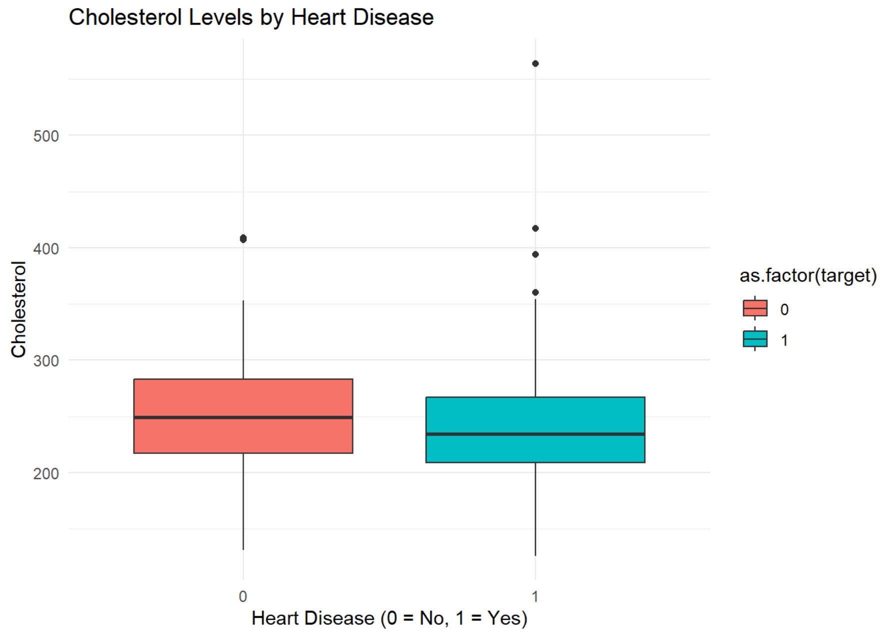


Step 4: Data Distribution Visualizations

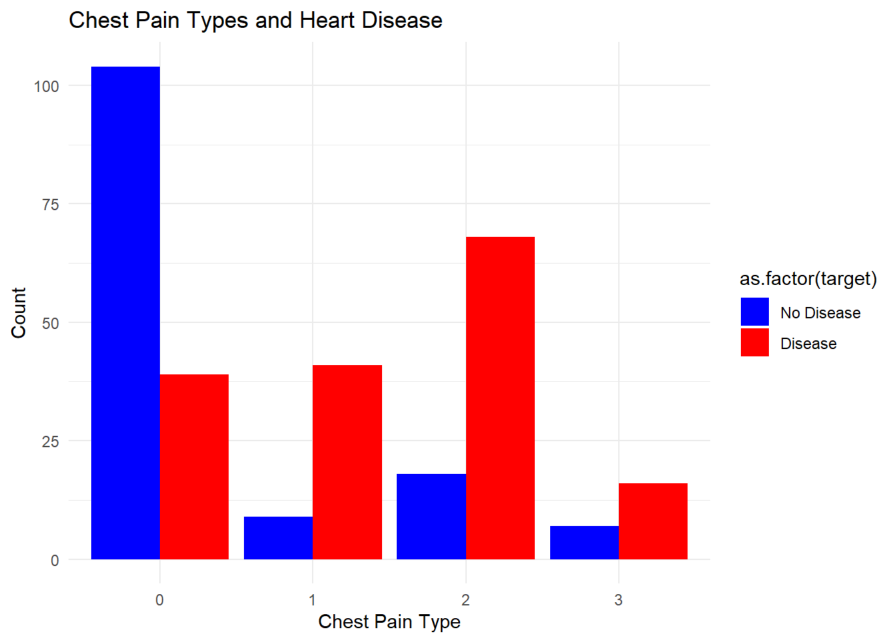
```
# A. Visualize Numerical Features
library(ggplot2)

# Histogram for Age
ggplot(heart, aes(x = age)) +
  geom_histogram(binwidth = 5, fill = "blue", color = "black") +
  labs(title = "Distribution of Age", x = "Age", y = "Count") +
  theme_minimal()
```



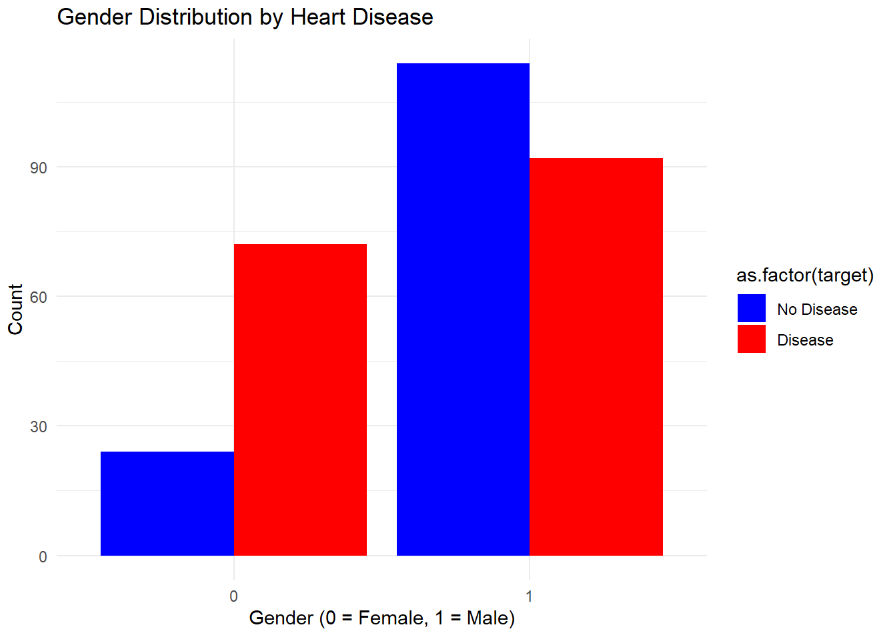
```
# Boxplot for Cholesterol by Target
ggplot(heart, aes(x = as.factor(target), y = chol, fill = as.factor(target))) +
  geom_boxplot() +
  labs(title = "Cholesterol Levels by Heart Disease", x = "Heart Disease (0 = No, 1 = Yes)", y = "Cholesterol") +
  theme_minimal()
```



```
# B. Bar Charts for Categorical Variables

# Bar Chart for Chest Pain Type
ggplot(heart, aes(x = as.factor(cp), fill = as.factor(target))) +
  geom_bar(position = "dodge") +
  labs(title = "Chest Pain Types and Heart Disease", x = "Chest Pain Type", y = "Count") +
  scale_fill_manual(values = c("blue", "red"), labels = c("No Disease", "Disease")) +
  theme_minimal()
```



```
# Gender Distribution
ggplot(heart, aes(x = as.factor(sex), fill = as.factor(target))) +
  geom_bar(position = "dodge") +
  labs(title = "Gender Distribution by Heart Disease", x = "Gender (0 = Female, 1 = Male)", y = "Count") +
  scale_fill_manual(values = c("blue", "red"), labels = c("No Disease", "Disease")) +
  theme_minimal()
```

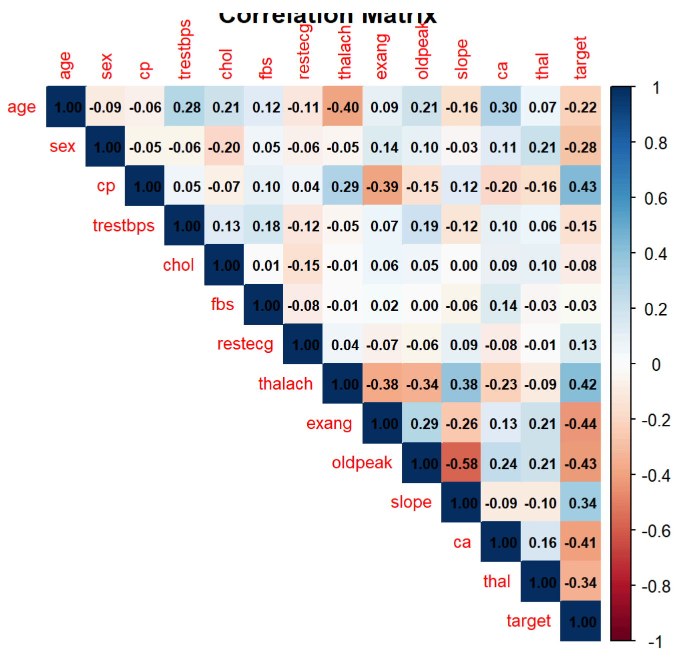


```r
# C. Visualize Relationships Between Numerical Variables
library(corrplot)
```

```
## Warning: package 'corrplot' was built under R version 4.3.3
```

```
## corrplot 0.95 loaded
```

```r
# Compute Correlation Matrix
numeric_data <- heart[, sapply(heart, is.numeric)]
cor_matrix <- cor(numeric_data)

# Plot Correlation Heatmap
corrplot(cor_matrix, method = "color", type = "upper",
         addCoef.col = "black", tl.cex = 0.8, number.cex = 0.7,
         main = "Correlation Matrix")
```



	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
age	1.00	-0.09	-0.06	0.28	0.21	0.12	-0.11	-0.40	0.09	0.21	-0.16	0.30	0.07	-0.22
sex		1.00	-0.05	-0.06	-0.20	0.05	-0.06	-0.05	0.14	0.10	-0.03	0.11	0.21	-0.28
cp			1.00	0.05	-0.07	0.10	0.04	0.29	-0.39	-0.15	0.12	-0.20	-0.16	0.43
trestbps				1.00	0.13	0.18	-0.12	-0.05	0.07	0.19	-0.12	0.10	0.06	-0.15
chol					1.00	0.01	-0.15	-0.01	0.06	0.05	0.00	0.09	0.10	-0.08
fbs						1.00	-0.08	-0.01	0.02	0.00	-0.06	0.14	-0.03	-0.03
restecg							1.00	0.04	-0.07	-0.06	0.09	-0.08	-0.01	0.13
thalach								1.00	-0.38	-0.34	0.38	-0.23	-0.09	0.42
exang									1.00	0.29	-0.26	0.13	0.21	-0.44
oldpeak										1.00	-0.58	0.24	0.21	-0.43
slope											1.00	-0.09	-0.10	0.34
ca												1.00	0.16	-0.41
thal													1.00	-0.34
target														1.00

Step 5: Data Transformation (Converting Variables into Factors)

1. Some variables represent categories, not quantities. Treating them as numbers would mislead the model
2. Conversion is important Logistic regression and other models can handle categorical data better when it is encoded as factors. This improves prediction accuracy and Factors make the data more understandable for humans and machines. For instance, sex = Female/Male is clearer than 0/1

```
heart$sex <- as.factor(heart$sex)
heart$cp <- as.factor(heart$cp)
heart$fbs <- as.factor(heart$fbs)
heart$restecg <- as.factor(heart$restecg)
heart$exang <- as.factor(heart$exang)
heart$slope <- as.factor(heart$slope)
heart$ca <- as.factor(heart$ca)
heart$thal <- as.factor(heart$thal)
heart$target <- as.factor(heart$target)
```

Step 6: Splitting Data (This step involves dividing the dataset into two parts: training data (80%) and testing data(20%))

Why Split the Data? 1. Training Data (80%): Used to train (build) the model by finding patterns in the data 2. Testing Data (20%): Used to evaluate the model’s accuracy and performance on new, unseen data 3. Reason: This mimics real-world scenarios where the model is applied to data it hasn’t seen before, preventing over fitting (when a model performs well on training data but poorly on new data)

Why Not Use the Entire Data set for Training? 1. Risk of Over fitting: The model might “memorize” the training data, performing exceptionally well on it but failing to generalize to new data 2. Real-World Simulation: In real-life applications, you need the model to work on data it hasn’t seen before

```
library(caTools)
```

```
## Warning: package 'caTools' was built under R version 4.3.3
```

```
set.seed(2) #to ensure reproducibility. Without this, every time you split the data, you'll get different results
split <- sample.split(heart$target, SplitRatio = 0.8)
training <- subset(heart, split == TRUE)
testing <- subset(heart, split == FALSE)
```

Step 7: Building the model (This step involves using the training data to create a mathematical model that predicts the target variable (e.g., presence of heart disease))

1. Here we used logistic regression model because the target variable (target) is binary (0 = no heart disease, 1 = heart disease)

```
model <- glm(target~., training, family = binomial)
summary(model)
```

```
##
## Call:
## glm(formula = target ~ ., family = binomial, data = training)
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.295200   4.353818   0.068 0.945943
## age          0.016480   0.030258   0.545 0.585989
## sex1        -2.624782   0.747849  -3.510 0.000448 ***
## cp1          1.306173   0.714180   1.829 0.067413 .
## cp2          2.596896   0.671185   3.869 0.000109 ***
## cp3          2.920285   0.861528   3.390 0.000700 ***
## trestbps    -0.035263   0.013776  -2.560 0.010474 *
## chol        -0.006496   0.005081  -1.278 0.201115
## fbs1         1.341979   0.831161   1.615 0.106401
## restecg1     0.074422   0.478785   0.155 0.876474
## restecg2    -0.243861   3.353627  -0.073 0.942032
## thalach      0.033606   0.017009   1.976 0.048174 *
## exang1      -0.619649   0.545508  -1.136 0.255993
## oldpeak     -0.409794   0.267555  -1.532 0.125615
## slope1      -0.893167   1.000572  -0.893 0.372041
## slope2       0.861472   1.072122   0.804 0.421674
## ca1         -2.077715   0.649907  -3.197 0.001389 **
## ca2         -3.701530   0.989076  -3.742 0.000182 ***
## ca3         -1.845235   1.123495  -1.642 0.100506
## ca4         -1.084200   3.504617  -0.309 0.757045
## thal1        3.856076   2.968733   1.299 0.193980
## thal2        2.734698   2.856244   0.957 0.338342
## thal3        1.295089   2.852677   0.454 0.649835
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 332.26  on 240  degrees of freedom
## Residual deviance: 126.82  on 218  degrees of freedom
## AIC: 172.82
##
## Number of Fisher Scoring iterations: 7
```

Interpreting the Results 1. Odds Ratio: For each unit increase in a variable, the odds of heart disease change by the factor e^β (exponent of the coefficient). 1.1 Example: If cp2 has a coefficient of 2.85, then e^2.85≈ 17.3. This means cp2 increases the odds of heart disease 17-fold compared to the baseline.

Step 8: Model optimization (Model Optimization involves improving the logistic regression model by removing insignificant or redundant variables. This step ensures the model is efficient, interpretable, and performs better on unseen data)

Purpose of Model Optimization 1. Simplify the Model: Reduce complexity by removing unnecessary predictors 2. Improve Performance: Focus on significant predictors to enhance accuracy 3. Lower AIC (Akaike Information Criterion): A lower AIC indicates a better model with an optimal balance of fit and complexity

```
model <- glm(target~.-age, training, family = binomial)
summary(model)
```

```
##
## Call:
## glm(formula = target ~ . - age, family = binomial, data = training)
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  1.325665   4.011251   0.330 0.741032
## sex1        -2.645056   0.744208  -3.554 0.000379 ***
## cp1          1.314534   0.712747   1.844 0.065136 .
## cp2          2.614185   0.672591   3.887 0.000102 ***
## cp3          2.879652   0.851004   3.384 0.000715 ***
## trestbps    -0.033395   0.013231  -2.524 0.011602 *
## chol        -0.006145   0.004981  -1.234 0.217310
## fbs1         1.315863   0.823644   1.598 0.110129
## restecg1     0.072474   0.479066   0.151 0.879754
## restecg2    -0.203098   3.107969  -0.065 0.947897
## thalach      0.030176   0.015683   1.924 0.054334 .
## exang1      -0.628631   0.543178  -1.157 0.247142
## oldpeak     -0.423506   0.266105  -1.591 0.111497
## slope1      -0.879646   1.000413  -0.879 0.379248
## slope2       0.874920   1.071971   0.816 0.414398
## ca1         -2.029203   0.643416  -3.154 0.001612 **
## ca2         -3.555466   0.944992  -3.762 0.000168 ***
## ca3         -1.760305   1.124240  -1.566 0.117402
## ca4         -1.173104   3.359677  -0.349 0.726960
## thal1        3.911792   3.091294   1.265 0.205720
## thal2        2.791222   2.982312   0.936 0.349311
## thal3        1.356410   2.977998   0.455 0.648766
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 332.26  on 240  degrees of freedom
## Residual deviance: 127.12  on 219  degrees of freedom
## AIC: 171.12
##
## Number of Fisher Scoring iterations: 7
```

```
# Remove Insignificant Variables
  # Remove one variable at a time, starting with the least significant
  # Example: If age has a high p-value, it may not significantly predict heart disease in this dataset.
  # Removing it reduces AIC, improving model efficiency

      model <- glm(target~.-chol, training, family = binomial)
      summary(model)
```

```
##
## Call:
## glm(formula = target ~ . - chol, family = binomial, data = training)
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.44404    4.24888  -0.105 0.916766
## age          0.01272    0.02959   0.430 0.667287
## sex1        -2.41489    0.71785  -3.364 0.000768 ***
## cp1          1.22038    0.70464   1.732 0.083288 .
## cp2          2.58580    0.66586   3.883 0.000103 ***
## cp3          2.87907    0.85219   3.378 0.000729 ***
## trestbps    -0.03508    0.01383  -2.537 0.011182 *
## fbs1         1.31623    0.81371   1.618 0.105757
## restecg1     0.19375    0.46605   0.416 0.677608
## restecg2    -0.06578    3.01838  -0.022 0.982614
## thalach      0.02989    0.01624   1.840 0.065696 .
## exang1      -0.63343    0.53583  -1.182 0.237144
## oldpeak     -0.42392    0.26841  -1.579 0.114249
## slope1      -0.97229    1.00300  -0.969 0.332354
## slope2       0.78275    1.07296   0.730 0.465681
## ca1         -2.09718    0.65763  -3.189 0.001428 **
## ca2         -3.62163    0.98329  -3.683 0.000230 ***
## ca3         -1.83598    1.10052  -1.668 0.095261 .
## ca4         -1.08181    3.47259  -0.312 0.755400
## thal1        3.75233    2.93266   1.279 0.200723
## thal2        2.56257    2.81508   0.910 0.362663
## thal3        1.02733    2.80375   0.366 0.714057
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 332.26  on 240  degrees of freedom
## Residual deviance: 128.37  on 219  degrees of freedom
## AIC: 172.37
##
## Number of Fisher Scoring iterations: 7
```

```
#Chol should be removed since it reduces aic after removal.

model <- glm(target~.-fbs, training, family = binomial)
summary(model)
```

```
##
## Call:
## glm(formula = target ~ . - fbs, family = binomial, data = training)
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.707565   4.920003   0.144 0.885647
## age          0.014721   0.030413   0.484 0.628355
## sex1        -2.572096   0.747487  -3.441 0.000580 ***
## cp1          1.209278   0.694549   1.741 0.081666 .
## cp2          2.629193   0.654316   4.018 5.86e-05 ***
## cp3          3.003895   0.864801   3.474 0.000514 ***
## trestbps    -0.032134   0.013221  -2.431 0.015075 *
## chol        -0.006386   0.004972  -1.284 0.199035
## restecg1     0.125134   0.474661   0.264 0.792066
## restecg2    -0.301995   3.265169  -0.092 0.926309
## thalach      0.033734   0.016897   1.996 0.045886 *
## exang1      -0.651458   0.544731  -1.196 0.231725
## oldpeak     -0.436764   0.266089  -1.641 0.100710
## slope1      -0.968080   1.006197  -0.962 0.335990
## slope2       0.679868   1.077629   0.631 0.528111
## ca1         -1.906226   0.618536  -3.082 0.002057 **
## ca2         -3.514953   0.973282  -3.611 0.000304 ***
## ca3         -1.524615   1.046876  -1.456 0.145297
## ca4         -0.522147   4.651809  -0.112 0.910628
## thal1        3.303207   3.755174   0.880 0.379054
## thal2        2.051047   3.663348   0.560 0.575559
## thal3        0.705558   3.670620   0.192 0.847572
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 332.26  on 240  degrees of freedom
## Residual deviance: 129.54  on 219  degrees of freedom
## AIC: 173.54
##
## Number of Fisher Scoring iterations: 7
```

```
#fbs should not be removed since it increases aic after removal

model <- glm(target~.-restecg, training, family = binomial)
summary(model)
```

```
##
## Call:
## glm(formula = target ~ . - restecg, family = binomial, data = training)
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.363025   4.353294   0.083 0.933541
## age          0.016401   0.030232   0.543 0.587459
## sex1        -2.635390   0.740914  -3.557 0.000375 ***
## cp1          1.315717   0.712078   1.848 0.064644 .
## cp2          2.597341   0.671275   3.869 0.000109 ***
## cp3          2.924994   0.859950   3.401 0.000671 ***
## trestbps    -0.035573   0.013639  -2.608 0.009104 **
## chol        -0.006644   0.004961  -1.339 0.180501
## fbs1         1.346928   0.827585   1.628 0.103622
## thalach      0.033933   0.016907   2.007 0.044748 *
## exang1      -0.621461   0.545736  -1.139 0.254804
## oldpeak     -0.410861   0.265914  -1.545 0.122325
## slope1      -0.893861   0.996916  -0.897 0.369919
## slope2       0.870259   1.066995   0.816 0.414719
## ca1         -2.085554   0.647452  -3.221 0.001277 **
## ca2         -3.706346   0.986647  -3.757 0.000172 ***
## ca3         -1.848735   1.127038  -1.640 0.100933
## ca4         -1.088224   3.450364  -0.315 0.752463
## thal1        3.878334   3.008006   1.289 0.197281
## thal2        2.736131   2.900255   0.943 0.345471
## thal3        1.308902   2.894590   0.452 0.651133
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 332.26  on 240  degrees of freedom
## Residual deviance: 126.85  on 220  degrees of freedom
## AIC: 168.85
##
## Number of Fisher Scoring iterations: 7
```

```
#restecg should be removed since it reduces aic after removal

model <- glm(target~.-exang, training, family = binomial)
summary(model)
```

```
## 
## Call:
## glm(formula = target ~ . - exang, family = binomial, data = training)
## 
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)    
## (Intercept) -0.451644   4.102964  -0.110 0.912348    
## age          0.017589   0.029896   0.588 0.556301    
## sex1        -2.566693   0.733236  -3.501 0.000464 ***
## cp1          1.525123   0.688089   2.216 0.026660 *  
## cp2          2.853317   0.637938   4.473 7.72e-06 ***
## cp3          3.060928   0.857519   3.570 0.000358 ***
## trestbps    -0.035481   0.013831  -2.565 0.010308 *  
## chol        -0.006545   0.004967  -1.318 0.187579    
## fbs1         1.348230   0.813749   1.657 0.097557 .  
## restecg1     0.083141   0.476601   0.174 0.861516    
## restecg2    -0.215391   3.169351  -0.068 0.945817    
## thalach      0.035485   0.016760   2.117 0.034239 *  
## oldpeak     -0.445472   0.267789  -1.664 0.096208 .  
## slope1      -0.892251   0.989585  -0.902 0.367248    
## slope2       0.892124   1.061350   0.841 0.400597    
## ca1         -2.087751   0.649074  -3.217 0.001298 ** 
## ca2         -3.598974   0.974396  -3.694 0.000221 ***
## ca3         -1.855011   1.117170  -1.660 0.096823 .  
## ca4         -1.238432   3.346158  -0.370 0.711304    
## thal1        3.987878   2.734565   1.458 0.144752    
## thal2        2.843159   2.615743   1.087 0.277063    
## thal3        1.344503   2.609336   0.515 0.606367    
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## (Dispersion parameter for binomial family taken to be 1)
## 
##     Null deviance: 332.26  on 240  degrees of freedom
## Residual deviance: 128.10  on 219  degrees of freedom
## AIC: 172.1
## 
## Number of Fisher Scoring iterations: 7
```

```
#exang should be removed since it reduces aic after removal

model <- glm(target~.-oldpeak, training, family = binomial)
summary(model)
```

```
## 
## Call:
## glm(formula = target ~ . - oldpeak, family = binomial, data = training)
## 
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)    
## (Intercept) -0.381717   4.232311  -0.090 0.928135    
## age          0.021295   0.030209   0.705 0.480861    
## sex1        -2.711512   0.727086  -3.729 0.000192 ***
## cp1          1.406083   0.718866   1.956 0.050468 .  
## cp2          2.398675   0.638389   3.757 0.000172 ***
## cp3          2.734715   0.858382   3.186 0.001443 ** 
## trestbps    -0.036830   0.013751  -2.678 0.007401 ** 
## chol        -0.006730   0.005041  -1.335 0.181857    
## fbs1         1.367347   0.790969   1.729 0.083863 .  
## restecg1     0.082123   0.473702   0.173 0.862365    
## restecg2    -0.600034   2.904674  -0.207 0.836342    
## thalach      0.034541   0.016853   2.049 0.040415 *  
## exang1      -0.708292   0.538294  -1.316 0.188239    
## slope1      -0.507411   0.901779  -0.563 0.573654    
## slope2       1.501021   0.926555   1.620 0.105232    
## ca1         -2.010400   0.639365  -3.144 0.001664 ** 
## ca2         -3.909845   0.959312  -4.076 4.59e-05 ***
## ca3         -1.944451   1.082769  -1.796 0.072524 .  
## ca4         -0.666123   3.643758  -0.183 0.854945    
## thal1        3.552278   2.822071   1.259 0.208121    
## thal2        2.589238   2.710921   0.955 0.339520    
## thal3        1.103418   2.708624   0.407 0.683735    
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## (Dispersion parameter for binomial family taken to be 1)
## 
##     Null deviance: 332.26  on 240  degrees of freedom
## Residual deviance: 129.31  on 219  degrees of freedom
## AIC: 173.31
## 
## Number of Fisher Scoring iterations: 7
```

```
#oldpeak should not be removed since it increases aic after removal

model <- glm(target~.-slope, training, family = binomial)
summary(model)
```

```
##
## Call:
## glm(formula = target ~ . - slope, family = binomial, data = training)
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.742790   3.584117  -0.207 0.835818
## age          0.015372   0.030434   0.505 0.613505
## sex1        -2.211146   0.676249  -3.270 0.001077 **
## cp1          1.347198   0.686410   1.963 0.049684 *
## cp2          2.397706   0.625744   3.832 0.000127 ***
## cp3          2.655163   0.806184   3.293 0.000990 ***
## trestbps    -0.033253   0.013236  -2.512 0.011994 *
## chol        -0.006654   0.004919  -1.353 0.176135
## fbs1         1.030259   0.764565   1.348 0.177816
## restecg1     0.186206   0.464385   0.401 0.688440
## restecg2    -0.357638   3.282078  -0.109 0.913229
## thalach      0.040990   0.015769   2.599 0.009337 **
## exang1      -0.695550   0.532766  -1.306 0.191708
## oldpeak     -0.636878   0.253941  -2.508 0.012143 *
## ca1         -1.617110   0.584633  -2.766 0.005674 **
## ca2         -2.934552   0.873648  -3.359 0.000782 ***
## ca3         -1.492465   0.994232  -1.501 0.133324
## ca4         -1.135837   2.763365  -0.411 0.681048
## thal1        3.130701   2.100989   1.490 0.136196
## thal2        2.371865   1.949207   1.217 0.223667
## thal3        0.809129   1.946088   0.416 0.677577
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 332.26  on 240  degrees of freedom
## Residual deviance: 136.30  on 220  degrees of freedom
## AIC: 178.3
##
## Number of Fisher Scoring iterations: 6
```

```
#slope should not be removed since it reduces aic after removal

model <- glm(target~.-thal, training, family = binomial)
summary(model)
```

```
##
## Call:
## glm(formula = target ~ . - thal, family = binomial, data = training)
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  4.609153   3.473813   1.327 0.184566
## age          0.012875   0.029076   0.443 0.657895
## sex1        -2.873978   0.637989  -4.505 6.65e-06 ***
## cp1          1.608956   0.698737   2.303 0.021298 *
## cp2          2.507789   0.615335   4.075 4.59e-05 ***
## cp3          2.777243   0.775936   3.579 0.000345 ***
## trestbps    -0.032936   0.013123  -2.510 0.012077 *
## chol        -0.009421   0.004888  -1.928 0.053912 .
## fbs1         1.075935   0.724441   1.485 0.137492
## restecg1    -0.054437   0.450792  -0.121 0.903883
## restecg2    -0.729176   2.797157  -0.261 0.794336
## thalach      0.026316   0.014890   1.767 0.077164 .
## exang1      -0.868600   0.527297  -1.647 0.099503 .
## oldpeak     -0.392701   0.244089  -1.609 0.107651
## slope1      -0.978026   0.932975  -1.048 0.294506
## slope2       0.769392   1.031594   0.746 0.455771
## ca1         -1.942310   0.589225  -3.296 0.000979 ***
## ca2         -3.318062   0.894370  -3.710 0.000207 ***
## ca3         -2.177570   1.060378  -2.054 0.040016 *
## ca4         -0.998000   2.652173  -0.376 0.706698
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 332.26  on 240  degrees of freedom
## Residual deviance: 139.70  on 221  degrees of freedom
## AIC: 179.7
##
## Number of Fisher Scoring iterations: 6
```

```
      #Thal should not be removed since it increases aic after removal

#Final model after optimization
model <- glm(target~.-age-chol-restecg-exang, training, family = binomial)
summary(model)
```

```
##
## Call:
## glm(formula = target ~ . - age - chol - restecg - exang, family = binomial,
##     data = training)
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.17499    3.65481  -0.048 0.961812
## sex1        -2.39086    0.69210  -3.454 0.000551 ***
## cp1          1.46593    0.67782   2.163 0.030562 *
## cp2          2.85714    0.63490   4.500 6.79e-06 ***
## cp3          2.98425    0.83328   3.581 0.000342 ***
## trestbps    -0.03439    0.01311  -2.623 0.008729 **
## fbs1         1.30984    0.78043   1.678 0.093277 .
## thalach      0.02941    0.01465   2.008 0.044618 *
## oldpeak     -0.47145    0.26303  -1.792 0.073080 .
## slope1      -0.96292    0.97900  -0.984 0.325326
## slope2       0.83988    1.05150   0.799 0.424442
## ca1         -2.08207    0.64692  -3.218 0.001289 **
## ca2         -3.38838    0.91279  -3.712 0.000206 ***
## ca3         -1.80585    1.09189  -1.654 0.098153 .
## ca4         -1.31700    3.06447  -0.430 0.667367
## thal1        3.97623    2.87657   1.382 0.166886
## thal2        2.73084    2.76286   0.988 0.322950
## thal3        1.14514    2.74812   0.417 0.676900
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 332.26  on 240  degrees of freedom
## Residual deviance: 130.19  on 223  degrees of freedom
## AIC: 166.19
##
## Number of Fisher Scoring iterations: 6
```

Step 9: Model Assumptions and Diagnostics (step ensures the logistic regression model is appropriate, reliable, and performs as expected. It involves checking statistical assumptions, diagnosing potential problems, and validating the model’s behavior)

Purpose of Checking Assumptions: Logistic regression has specific assumptions that need to be validated for the model to give meaningful and accurate predictions.

Key assumptions include: 1. Linearity of Log-Odds: The relationship between independent variables and the log-odds of the target variable is linear 2. Independence of Residuals: Residuals (errors) are independent of each other 3. No Multicollinearity: Independent variables are not excessively correlated 4. Goodness-of-Fit: The model adequately fits the data

```
# Install required libraries if not already installed
#install.packages("performance")
#install.packages("car")
#install.packages("see")
#install.packages("ggplot2")
#install.packages("pROC")
#install.packages("magrittr")
#install.packages("ggeffects")

# Load libraries for diagnostics
library(performance)   # For model checks (e.g., residuals, multicollinearity)
```

```
## Warning: package 'performance' was built under R version 4.3.3
```

```
library(car)           # For Durbin-Watson test
```

```
## Warning: package 'car' was built under R version 4.3.3
```

```
## Loading required package: carData
```

```
## Warning: package 'carData' was built under R version 4.3.3
```

```
##
## Attaching package: 'car'
```

```
## The following object is masked from 'package:dplyr':
##
##     recode
```

```
## The following object is masked from 'package:purrr':
##
##     some
```

```
library(see)           # For enhanced plotting
```

```
## Warning: package 'see' was built under R version 4.3.3
```

```
library(ggplot2)       # For visualization
library(pROC)          # For ROC curve and AUC
```

```
## Warning: package 'pROC' was built under R version 4.3.3
```

```
## Type 'citation("pROC")' for a citation.
```

```
##
## Attaching package: 'pROC'
```

```
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```
library(magrittr)  # For the pipe operator
```

```
## Warning: package 'magrittr' was built under R version 4.3.3
```

```
##
## Attaching package: 'magrittr'
```

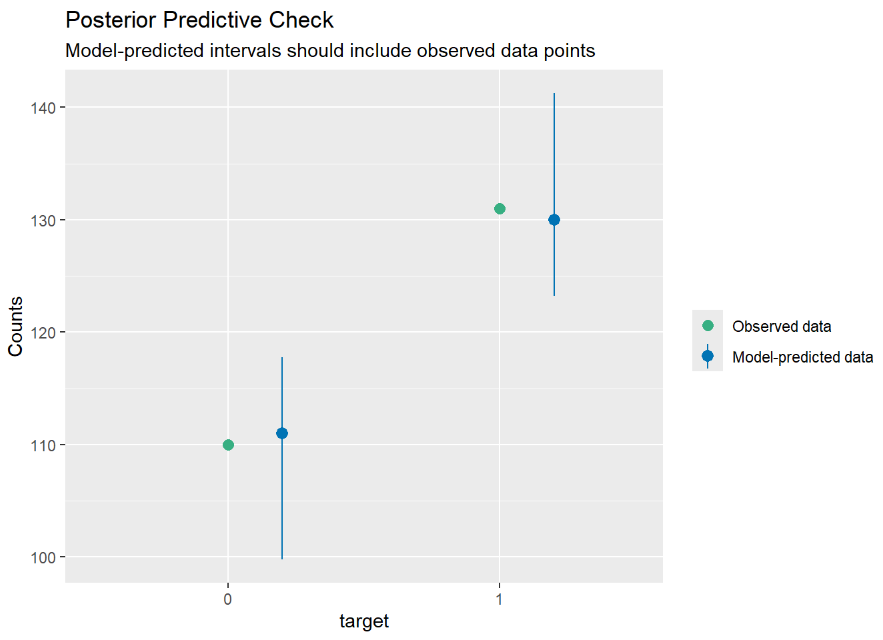
```
## The following object is masked from 'package:purrr':
##
##     set_names
```

```
## The following object is masked from 'package:tidyr':
##
##     extract
```

```
library(ggeffects) #
```

```
## Warning: package 'ggeffects' was built under R version 4.3.3
```

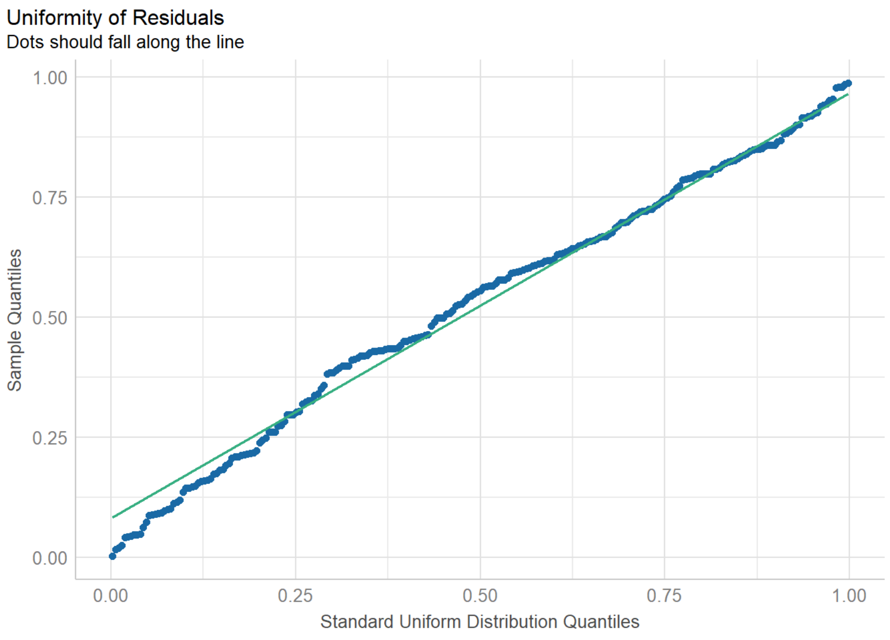
```
# A. Test of goodness of fit
# What to Check: Whether the model's predicted data aligns well with the observed data
# Why It Matters: A poorly fitting model may fail to generalize well to new data.
check_predictions(model)
```



```
# Result: The predicted data fits the observed data, so the assumption is not violated

# B. Check residuals (Linearity of Log-Odds)
# What to Check: Whether the relationship between predictors and the log-odds (logit) is linear
# Why It Matters: Non-linearity can lead to biased results. If the relationship is not linear, you might transform variables
or include interaction terms
check_residuals(model) %>% plot() # Assuming 'model' is your fitted model
```

```
## Warning in check_dep_version(): ABI version mismatch:
## lme4 was built with Matrix ABI version 1
## Current Matrix ABI version is 0
## Please re-install lme4 from source or restore original 'Matrix' package
```

```
## For confidence bands, please install `qqplotr`.
```



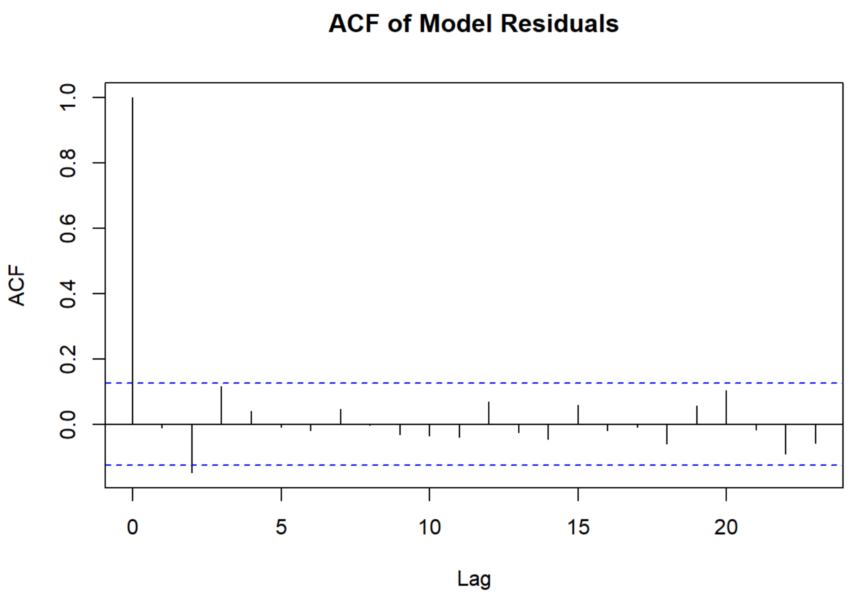
```
# C. Check for multicollinearity
# What to Check: Ensure independent variables are not highly correlated with each other
# Why It Matters: Multicollinearity can confuse the model about which variable is driving the prediction
check_collinearity(model)
```

```
## # Check for Multicollinearity
##
## Low Correlation
##
##      Term  VIF   VIF 95% CI Increased SE Tolerance Tolerance 95% CI
##       sex 1.80 [1.55, 2.16]         1.34      0.56     [0.46, 0.65]
##        cp 1.85 [1.59, 2.23]         1.36      0.54     [0.45, 0.63]
##  trestbps 1.19 [1.08, 1.44]         1.09      0.84     [0.70, 0.93]
##       fbs 1.23 [1.11, 1.48]         1.11      0.81     [0.68, 0.90]
##   thalach 1.43 [1.26, 1.71]         1.19      0.70     [0.59, 0.79]
##   oldpeak 1.45 [1.28, 1.74]         1.21      0.69     [0.58, 0.78]
##     slope 1.90 [1.63, 2.29]         1.38      0.53     [0.44, 0.61]
##        ca 2.11 [1.79, 2.55]         1.45      0.47     [0.39, 0.56]
##      thal 1.75 [1.51, 2.11]         1.32      0.57     [0.47, 0.66]
```

```
# Interpretation:
  # Variance Inflation Factor (VIF) values > 5 indicate problematic multicollinearity.
      # Example: In the file, VIF values are low (all < 2), suggesting no multicollinearity issues

# D. Test for autocorrelation (Independence of Residuals)
  # What to Check: Ensure residuals (differences between observed and predicted values) are independent
durbinWatsonTest(model)
```

```
##  lag Autocorrelation D-W Statistic p-value
##    1    -0.009517289      2.018653   0.912
##  Alternative hypothesis: rho != 0
```

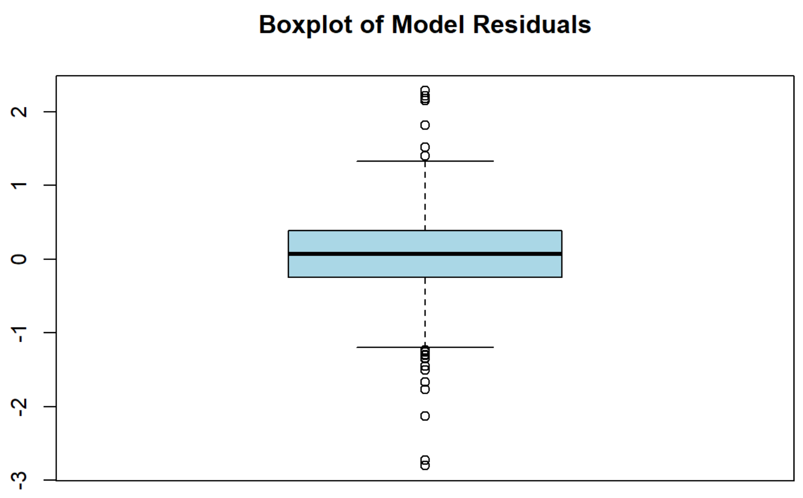
```
# Interpretation:
  # A p-value > 0.05 indicates no significant autocorrelation, meaning residuals are independent.
      #Example: In the file, residuals are independent (p = 0.94)

  # Plot the autocorrelation function (ACF)
  residuals_model <- residuals(model)
  acf(residuals_model, main = "ACF of Model Residuals")
```



```
# E. Outliers
# What to Check: Identify data points that significantly deviate from the model's expectations
outlierTest(model)
```

```
## No Studentized residuals with Bonferroni p < 0.05
## Largest |rstudent|:
##     rstudent unadjusted p-value Bonferroni p
## 34 -2.903318          0.0036923      0.88985
```

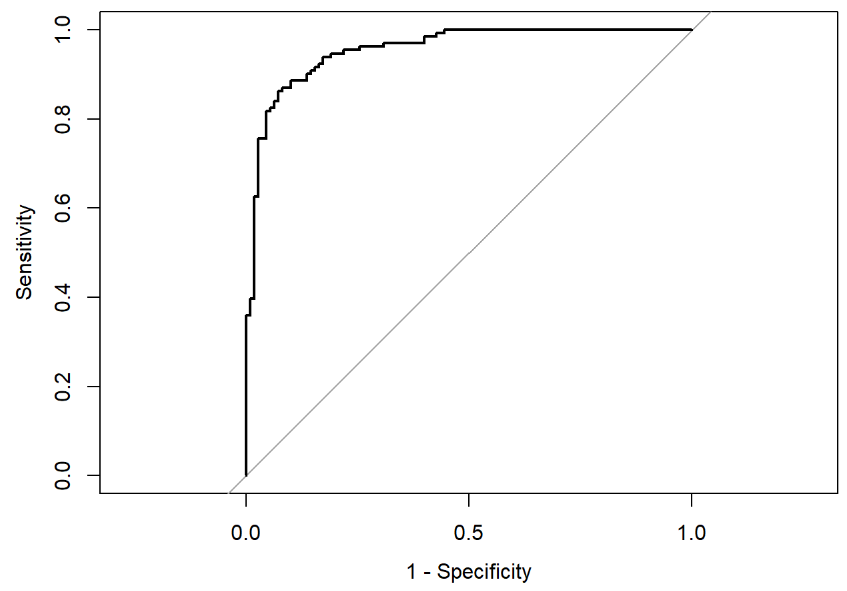
```
# Why It Matters: Outliers can distort model predictions, making them less reliable.
# Result: No significant outliers were detected

    # Boxplot of residuals to detect outliers
    residuals_model <- residuals(model)
    boxplot(residuals_model, main = "Boxplot of Model Residuals", col = "lightblue")
```



```
# F. Generate ROC curve
# Evaluates the model's ability to distinguish between classes (e.g., heart disease vs. no heart disease)
  roc(target ~ fitted.values(model), data = training, plot = TRUE, legacy.axes = TRUE)
```

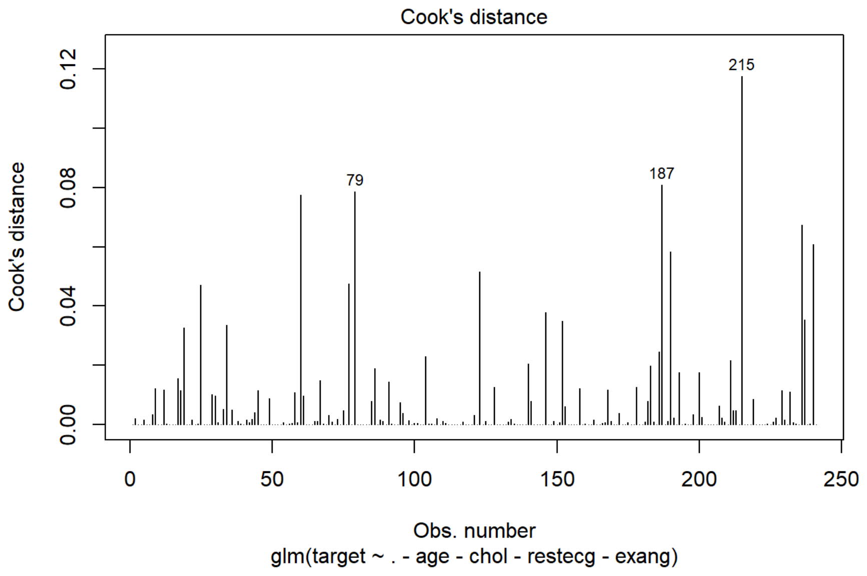
```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```



```
##
## Call:
## roc.formula(formula = target ~ fitted.values(model), data = training,     plot = TRUE, legacy.axes = TRUE)
##
## Data: fitted.values(model) in 110 controls (target 0) < 131 cases (target 1).
## Area under the curve: 0.9555
```

```
# Result: The data reports an AUC of 0.955, indicating excellent performance

# G. Cook's Distance: This diagnostic tool helps identify influential points in your data. If a point has a large Cook's Dis
tance, it indicates that the point has a significant effect on the model's parameters.
# plot(model, which = 4): This generates a plot of Cook's Distance for the fitted model (model), which can be useful for det
ecting influential observations.
plot(model, which = 4)
```



```
# How to Interpret the Plot:
# The plot will display the Cook's Distance for each observation.
# Points that are far away from the rest of the data (i.e., have large Cook's Distance values) are considered influential, a
nd you may want to investigate those points further.
```

Step 10: Predictions

1. predict() estimates the probability of heart disease for the test set.
2. Why: Predictions are categorized into risk levels (e.g., High Risk if probability > 0.6)

```
ggeffect(model)
```

```
## $age
## # Predicted probabilities of target
##
## age | Predicted |     95% CI
## ----------------------------
##  25 |      0.56 | 0.44, 0.67
##  30 |      0.56 | 0.44, 0.67
##  40 |      0.56 | 0.44, 0.67
##  45 |      0.56 | 0.44, 0.67
##  50 |      0.56 | 0.44, 0.67
##  60 |      0.56 | 0.44, 0.67
##  65 |      0.56 | 0.44, 0.67
##  80 |      0.56 | 0.44, 0.67
```

```
##
## Not all rows are shown in the output. Use `print(..., n = Inf)` to show
##   all rows.
```

```
##
## $sex
## # Predicted probabilities of target
##
## sex | Predicted |     95% CI
## ----------------------------
## 0   |      0.86 | 0.68, 0.95
## 1   |      0.37 | 0.24, 0.52
##
##
## $cp
## # Predicted probabilities of target
##
## cp | Predicted |     95% CI
## ---------------------------
## 0  |      0.25 | 0.15, 0.40
## 1  |      0.60 | 0.31, 0.83
## 2  |      0.86 | 0.69, 0.94
## 3  |      0.87 | 0.62, 0.96
##
##
## $trestbps
## # Predicted probabilities of target
##
## trestbps | Predicted |     95% CI
## ---------------------------------
##       94 |      0.82 | 0.62, 0.93
##      107 |      0.75 | 0.58, 0.87
##      121 |      0.65 | 0.52, 0.76
##      134 |      0.54 | 0.42, 0.65
##      147 |      0.43 | 0.29, 0.58
##      160 |      0.33 | 0.17, 0.54
##      173 |      0.24 | 0.09, 0.50
##      200 |      0.11 | 0.02, 0.43
```

```
##
## Not all rows are shown in the output. Use `print(..., n = Inf)` to show
##   all rows.
```

```
##
## $chol
## # Predicted probabilities of target
##
## chol | Predicted |     95% CI
## -----------------------------
##  120 |      0.56 | 0.44, 0.67
##  180 |      0.56 | 0.44, 0.67
##  240 |      0.56 | 0.44, 0.67
##  300 |      0.56 | 0.44, 0.67
##  340 |      0.56 | 0.44, 0.67
##  400 |      0.56 | 0.44, 0.67
##  460 |      0.56 | 0.44, 0.67
##  580 |      0.56 | 0.44, 0.67
```

```
##
## Not all rows are shown in the output. Use `print(..., n = Inf)` to show
##   all rows.
```

```
##
## $fbs
## # Predicted probabilities of target
##
## fbs | Predicted |     95% CI
## ----------------------------
## 0   |      0.51 | 0.39, 0.63
## 1   |      0.80 | 0.48, 0.94
##
##
## $restecg
## # Predicted probabilities of target
##
## restecg | Predicted |     95% CI
## --------------------------------
## 0       |      0.56 | 0.44, 0.67
## 1       |      0.56 | 0.44, 0.67
## 2       |      0.56 | 0.44, 0.67
##
##
## $thalach
## # Predicted probabilities of target
##
## thalach | Predicted |     95% CI
## --------------------------------
##      70 |      0.11 | 0.01, 0.57
##      85 |      0.16 | 0.03, 0.57
##     105 |      0.25 | 0.08, 0.58
##     120 |      0.35 | 0.16, 0.59
##     135 |      0.45 | 0.30, 0.61
##     155 |      0.60 | 0.48, 0.70
##     170 |      0.70 | 0.53, 0.82
##     205 |      0.87 | 0.56, 0.97
```

```
##
## Not all rows are shown in the output. Use `print(..., n = Inf)` to show
##   all rows.
```

```
##
## $exang
## # Predicted probabilities of target
##
## exang | Predicted |     95% CI
## ------------------------------
## 0     |      0.56 | 0.44, 0.67
## 1     |      0.56 | 0.44, 0.67
##
##
## $oldpeak
## # Predicted probabilities of target
##
## oldpeak | Predicted |     95% CI
## --------------------------------
##    0.00 |      0.67 | 0.51, 0.80
##    0.50 |      0.62 | 0.49, 0.73
##    1.50 |      0.50 | 0.37, 0.64
##    2.50 |      0.39 | 0.20, 0.62
##    3.00 |      0.33 | 0.14, 0.62
##    3.50 |      0.28 | 0.09, 0.62
##    4.50 |      0.20 | 0.04, 0.62
##    6.00 |      0.11 | 0.01, 0.63
```

```
##
## Not all rows are shown in the output. Use `print(..., n = Inf)` to show
##   all rows.
```

```
##
## $slope
## # Predicted probabilities of target
##
## slope | Predicted |     95% CI
## ------------------------------
## 0     |      0.57 | 0.17, 0.89
## 1     |      0.33 | 0.19, 0.51
## 2     |      0.75 | 0.58, 0.87
##
##
## $ca
## # Predicted probabilities of target
##
## ca | Predicted |     95% CI
## ---------------------------
## 0  |      0.78 | 0.65, 0.87
## 1  |      0.30 | 0.13, 0.55
## 2  |      0.11 | 0.02, 0.37
## 3  |      0.36 | 0.07, 0.81
## 4  |      0.48 | 0.00, 1.00
##
##
## $thal
## # Predicted probabilities of target
##
## thal | Predicted |     95% CI
## -----------------------------
## 0    |      0.13 | 0.00, 0.97
## 1    |      0.89 | 0.59, 0.98
## 2    |      0.69 | 0.54, 0.81
## 3    |      0.31 | 0.18, 0.50
##
##
## attr(,"class")
## [1] "ggalleffects" "list"
## attr(,"model.name")
## [1] "model"
```

```
ggpredict(model, terms = "sex")
```

```
## # Predicted probabilities of target
##
## sex | Predicted |     95% CI
## ----------------------------
## 0   |      0.31 | 0.00, 0.99
## 1   |      0.04 | 0.00, 0.93
##
## Adjusted for:
## *      age =  54.52
## *       cp =      0
## * trestbps = 131.91
## *     chol = 244.69
## *      fbs =      0
## *  restecg =      0
## *  thalach = 149.77
## *    exang =      0
## *  oldpeak =   1.03
## *    slope =      0
## *       ca =      0
## *     thal =      0
```

Step 11: Model Evaluation

1. The Model Evaluation step assesses the performance of the model using various metrics. These metrics provide insights into how well the model predicts outcomes and whether it generalizes well to unseen data

```
# A. Confusion Matrix
# What it is: A table comparing predicted values to actual values
predictions <- predict(model, testing, type = "response") # Generate predicted probabilities
classified <- ifelse(predictions > 0.5, 1, 0) # Convert probabilities to binary predictions
confusion_matrix <- table(Actual = testing$target, Predicted = classified) # Confusion matrix
print(confusion_matrix)
```

```
##       Predicted
## Actual  0  1
##      0 22  6
##      1  5 28
```

```
# Interpretation:
  # True Positives (TP): Model correctly predicts 1 (e.g., 28 cases).
  # True Negatives (TN): Model correctly predicts 0 (e.g., 22 cases).
  # False Positives (FP): Model incorrectly predicts 1 when it’s actually 0 (e.g., 6 cases).
  # False Negatives (FN): Model incorrectly predicts 0 when it’s actually 1 (e.g., 5 cases).

# B. Accuracy
# The accuracy of a model measures the proportion of correct predictions (both positives and negatives) out of all predictio
ns made
# Define confusion matrix values
TP <- 28
TN <- 22
FP <- 6
FN <- 5

# Calculate accuracy
accuracy <- (TP + TN) / (TP + TN + FP + FN) * 100
print(accuracy)
```

```
## [1] 81.96721
```

```
# Interpretation
  # An accuracy of 81.97% means that the model correctly predicts the target variable for approximately 82 out of 100 patien
ts.
  # While accuracy is a good general measure, it may not be sufficient in imbalanced datasets. For example, if the dataset h
as many more patients without heart disease than with it, the model might achieve high accuracy just by predicting "no heart
disease" for most cases

# C. Recall (Sensitivity)
# Recall (Sensitivity) measures the model's ability to correctly identify positive cases (e.g., patients with heart diseas
e). It answers the question: Of all the actual positives, how many did the model correctly predict as positive?
# Define confusion matrix values
TP <- 28
FN <- 5
# Calculate recall
recall <- (TP) / (TP + FN) * 100
print(recall)
```

```
## [1] 84.84848
```

```
# Interpretation
  # A recall of 84.85% means that the model correctly identifies about 85 out of 100 patients who actually have heart diseas
e.
  # High recall is especially important in situations where missing positive cases is costly, such as diagnosing diseases

# D. Precision
# Precision measures the model's ability to correctly identify positive predictions. It answers the question: Of all the cas
es the model predicted as positive, how many were actually positive?
# Define confusion matrix values
TP <- 28
FP <- 6
# Calculate precision
precision <- (TP) / (TP + FP) * 100
print(precision)
```

```
## [1] 82.35294
```

```
# Interpretation
  # A precision of 82.35% means that when the model predicts heart disease (positive), about 82 out of 100 predictions are c
orrect.
  # High precision is important in cases where false positives (incorrectly predicting someone has heart disease) are costl
y, such as in resource-intensive follow-up tests

# E. F1-Score
# The F1-Score is a metric that balances Precision and Recall. It provides a single score to evaluate a model’s performance
when there is a trade-off between these two metrics. It is especially useful for imbalanced datasets
# Calculate F1-Score
f1_score <- 2 * (precision * recall) / (precision + recall)
print(f1_score)
```

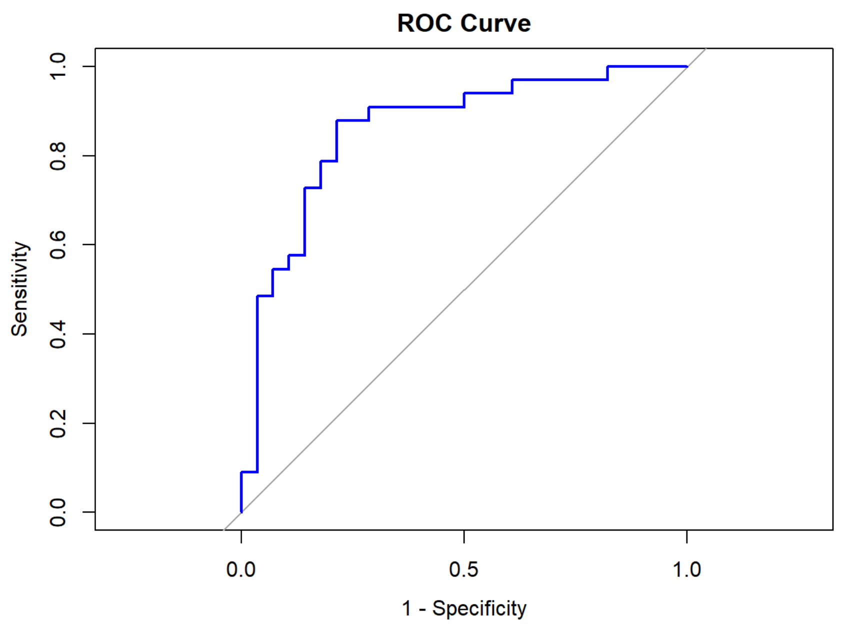
```
## [1] 83.58209
```

```
# Interpretation
  # The F1-Score of 83.58% indicates a good balance between Precision and Recall.
  # It means the model is effective in predicting both actual positives (Recall) and accurate positives (Precision)

# F. ROC (Receiver Operating Characteristic) Curve and AUC (Area Under the Curve)
# The ROC (Receiver Operating Characteristic) Curve and AUC (Area Under the Curve) are metrics used to evaluate the performa
nce of a classification model. They measure how well the model distinguishes between the two classes (e.g., heart disease pr
esent vs. not present).
roc_curve <- roc(testing$target, predictions)
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```
# Plot the ROC curve
plot(roc_curve, legacy.axes = TRUE, col = "blue", main = "ROC Curve", lwd = 2)
```



```
# Calculate the AUC
auc_value <- auc(roc_curve)
print(auc_value)
```

```
## Area under the curve: 0.8593
```

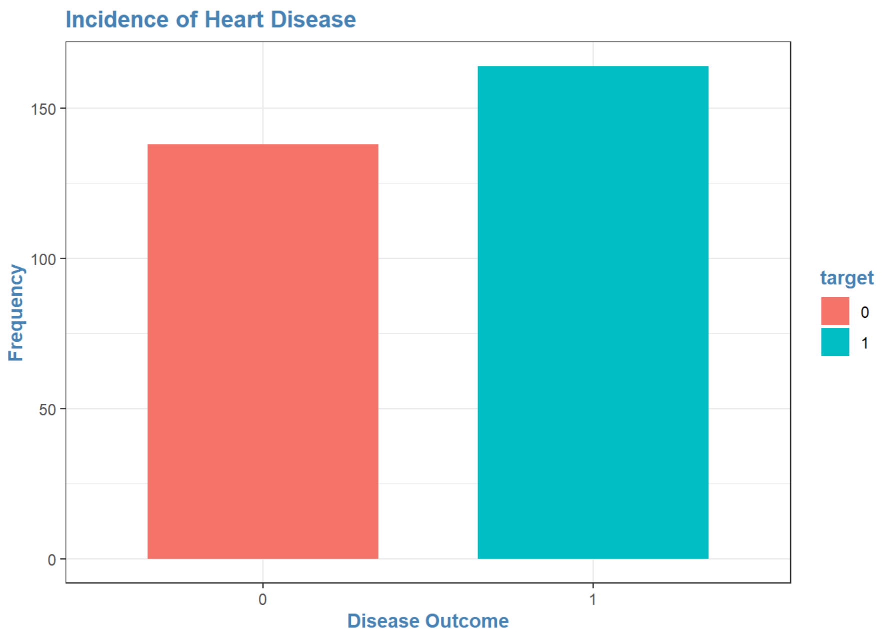
```
# Interpretation
  # AUC = 0.8723: The model has excellent discriminatory ability, meaning it can distinguish well between patients with and
without heart disease.
  # ROC Curve: A curve that rises steeply toward the top-left corner indicates a strong classifier.

# G. Calculate MCC
# The Matthews Correlation Coefficient (MCC) is a balanced metric for binary classification, especially useful for datasets
with imbalanced classes. It considers true and false positives and negatives, providing a correlation value between the pred
icted and actual classifications
mcc <- ((TP * TN) - (FP * FN)) / sqrt((TP + FP) * (TP + FN) * (TN + FP) * (TN + FN))
print(mcc)
```

```
## [1] 0.6362683
```

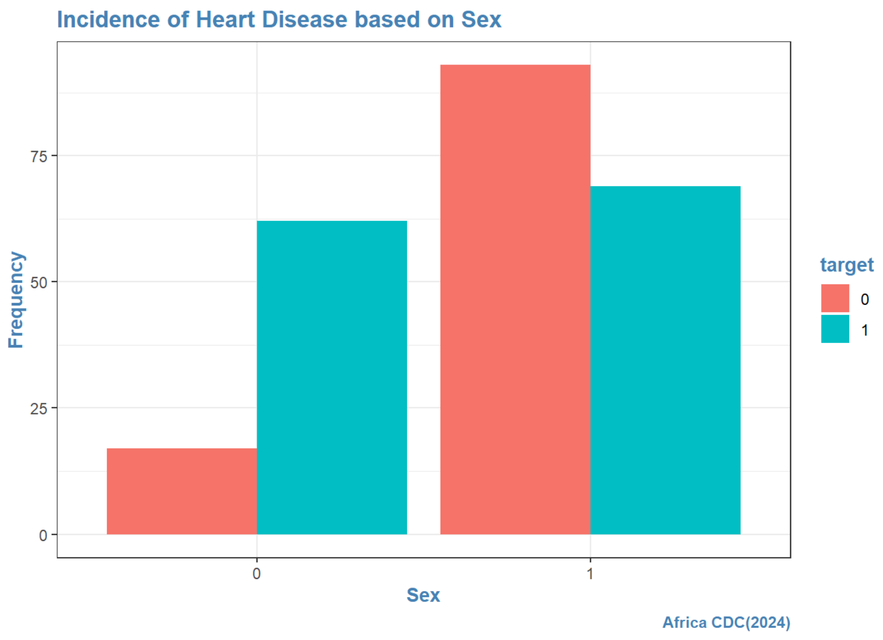
```
# Interpretation
  # MCC = 0.637: The model has good predictive ability.
  # MCC ranges from -1 to 1:
      # +1: Perfect prediction.
      # 0: No better than random guessing.
      # -1: Total disagreement between predictions and actual values.
```

Step 12: Visualize model predictions

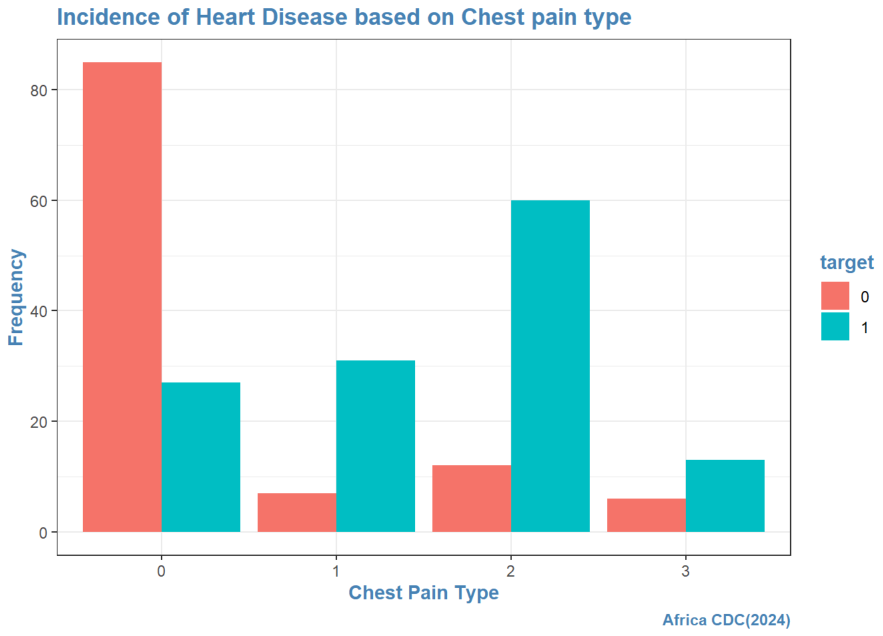
```
# A. Overall Incidence of Heart Disease
theme_set(theme_bw()+theme(title = element_text
                           (face = "bold",
                             colour = "steelblue")))
ggplot(heart, aes(target, fill = target))+
  geom_bar(width = 0.7)+
  labs(title = "Incidence of Heart Disease",
       y = "Frequency",
       x = "Disease Outcome")
```



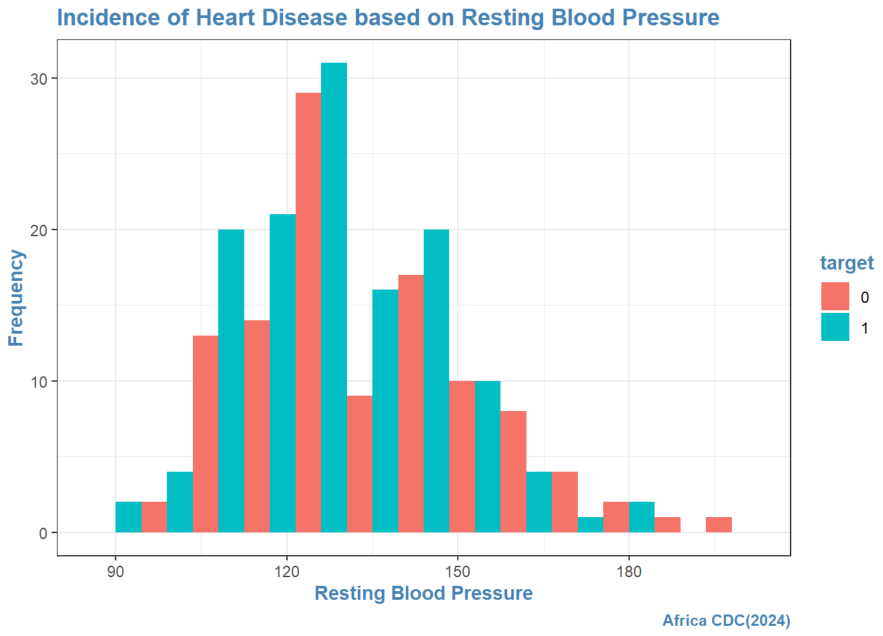
```
# B. Incidence of Heart Disease based on Sex
ggplot(training, aes(sex, fill = target))+
  geom_bar(position = "dodge")+
  labs(title = "Incidence of Heart Disease based on Sex",
       y = "Frequency",
       x = "Sex",
       caption = "Africa CDC(2024)")
```



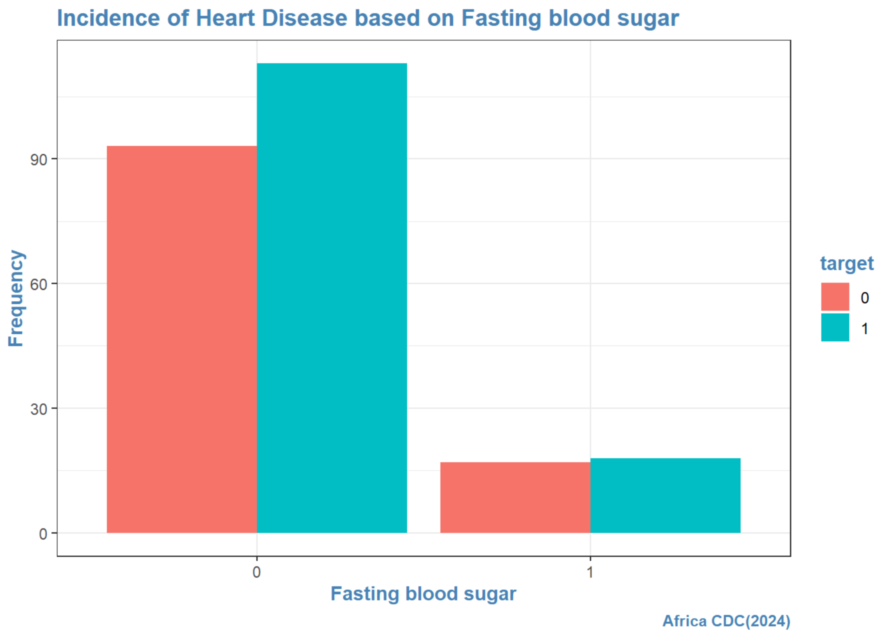
```
# C. Incidence of Heart Disease based on Chest pain type
ggplot(training, aes(cp, fill = target))+
  geom_bar(position = "dodge")+
  labs(title = "Incidence of Heart Disease based on Chest pain type",
       y = "Frequency",
       x = "Chest Pain Type",
       caption = "Africa CDC(2024)")
```



```
# D. Incidence of Heart Disease based on Resting Blood Pressure
ggplot(training, aes(trestbps, fill = target))+
  geom_histogram(binwidth = 9, position = "dodge")+
  labs(title = "Incidence of Heart Disease based on Resting Blood Pressure",
       y = "Frequency",
       x = "Resting Blood Pressure",
       caption = "Africa CDC(2024)")
```



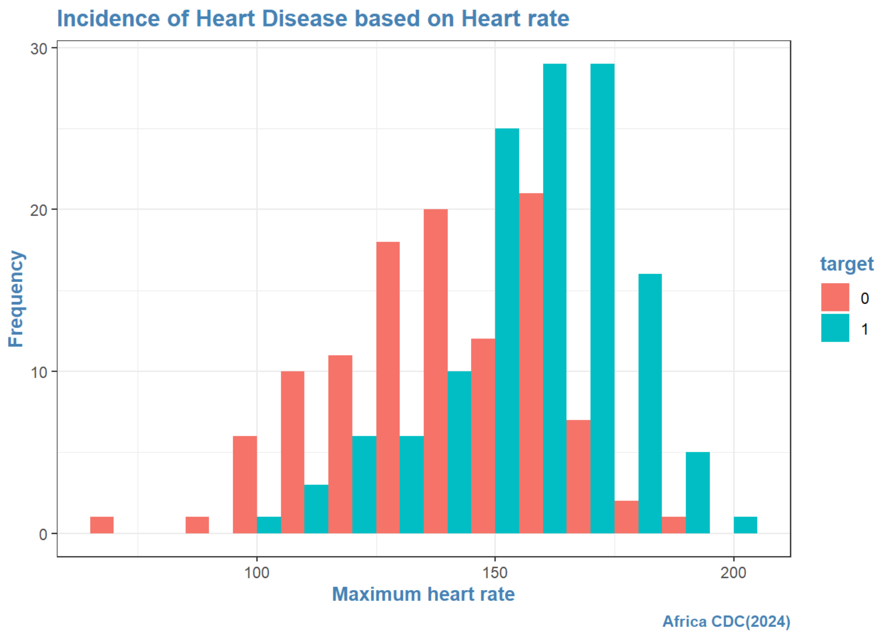
```
# E. Incidence of Heart Disease based on Fasting blood sugar
ggplot(training, aes(fbs, fill = target))+
  geom_bar(position = "dodge")+
  labs(title = "Incidence of Heart Disease based on Fasting blood sugar",
       y = "Frequency",
       x = "Fasting blood sugar",
       caption = "Africa CDC(2024)")
```



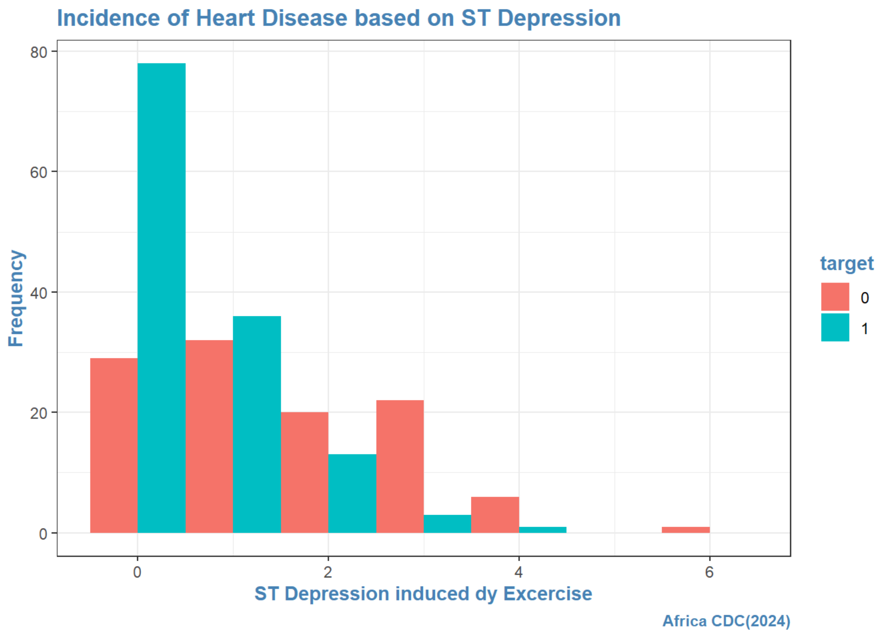
```
table(heart$target, heart$fbs)
```

```
##
##       0   1
##   0 116  22
##   1 141  23
```

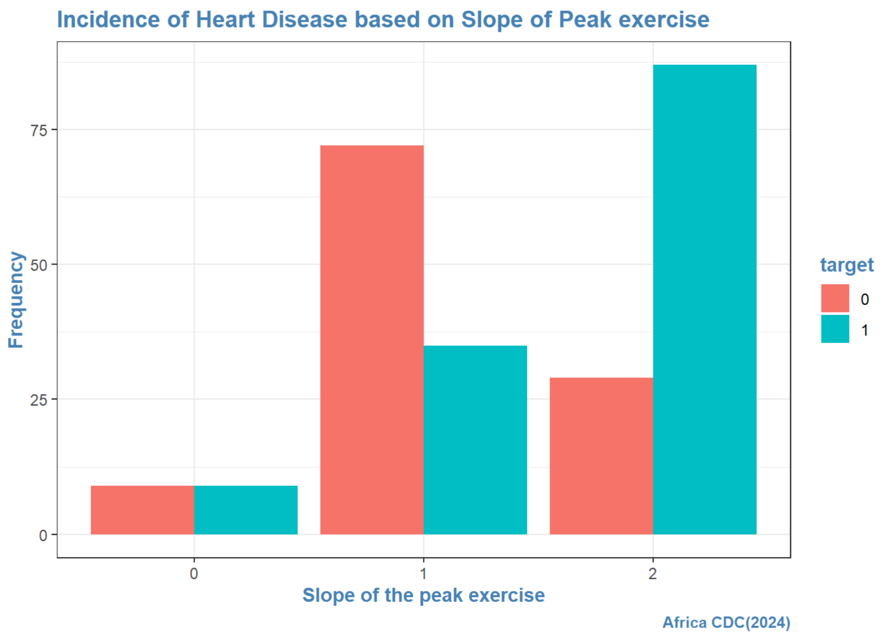
```
# F. Incidence of Heart Disease based on Heart rate
ggplot(training, aes(thalach, fill = target))+
  geom_histogram(binwidth = 10, position = "dodge")+
  labs(title = "Incidence of Heart Disease based on Heart rate",
       y = "Frequency",
       x = "Maximum heart rate",
       caption = "Africa CDC(2024)")
```



```
# G. Incidence of Heart Disease based on ST Depression
ggplot(training, aes(oldpeak, fill = target))+
  geom_histogram(binwidth = 1, position = "dodge")+
  labs(title = "Incidence of Heart Disease based on ST Depression",
       y = "Frequency",
       x = "ST Depression induced dy Excercise",
       caption = "Africa CDC(2024)")
```



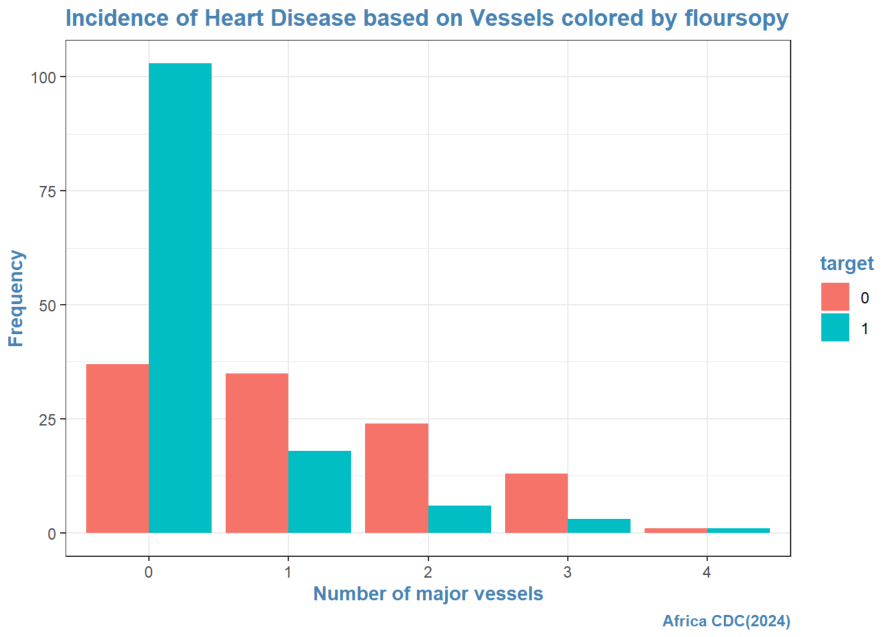
```
# H. Incidence of Heart Disease based on Slope of Peak exercise
ggplot(training, aes(slope, fill = target))+
  geom_bar(position = "dodge")+
  labs(title = "Incidence of Heart Disease based on Slope of Peak exercise",
       y = "Frequency",
       x = "Slope of the peak exercise",
       caption = "Africa CDC(2024)")
```



```
table(heart$target, heart$slope)
```

```
##
##       0   1   2
##   0  12  91  35
##   1   9  49 106
```

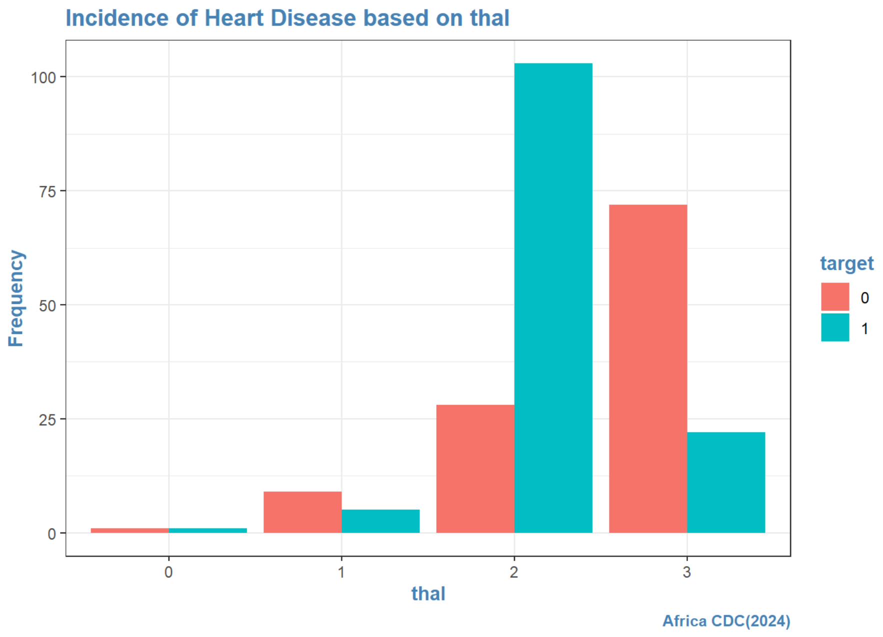
```
# I. Incidence of Heart Disease based on Vessels colored by floursopy
ggplot(training, aes(ca, fill = target))+
  geom_bar(position = "dodge")+
  labs(title = "Incidence of Heart Disease based on Vessels colored by floursopy",
       y = "Frequency",
       x = "Number of major vessels",
       caption = "Africa CDC(2024)")
```



```
table(heart$target, heart$ca)
```

```
## 
##       0   1   2   3   4
##   0  45  44  31  17   1
##   1 130  21   7   3   3
```

```
# J. Incidence of Heart Disease based on thal
ggplot(training, aes(thal, fill = target))+
  geom_bar(position = "dodge")+
  labs(title = "Incidence of Heart Disease based on thal",
       y = "Frequency",
       caption = "Africa CDC(2024)")
```



Step 13: Check Model Performance

```
# Produce model equation
#install.packages("equatiomatic")
library(equatiomatic)
```

```
## Warning: package 'equatiomatic' was built under R version 4.3.3
```

```
extract_eq(model)
```

$$\log\left[\frac{P(\text{target}=1)}{1-P(\text{target}=1)}\right] = \alpha + \beta_1(\text{sex}_1) + \beta_2(\text{cp}_1) + \beta_3(\text{cp}_2) + \beta_4(\text{cp}_3) + \beta_5(\text{trestbps}) + \beta_6(\text{fbs}_1) + \beta_7(\text{thalach}) + \beta_8(\text{oldpeak}) + \beta_9(\text{slop}$$

```
#How well our model fits the data
library(performance)
performance(model)
```

```
## # Indices of model performance
##
## AIC     |    AICc |     BIC | Tjur's R2 |  RMSE | Sigma | Log_loss | Score_log
## -----------------------------------------------------------------------------
## 166.193 | 169.274 | 228.920 |     0.665 | 0.287 | 1.000 |    0.270 |      -Inf
##
## AIC     | Score_spherical |   PCP
## ---------------------------------
## 166.193 |           0.025 | 0.834
```

```
library(effectsize)
```

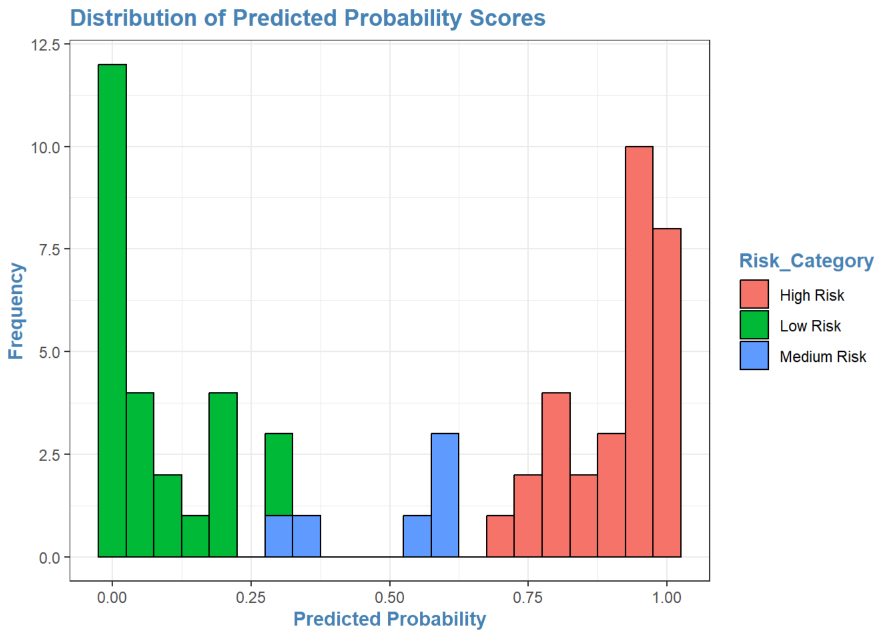
```
## Warning: package 'effectsize' was built under R version 4.3.3
```

```
interpret_r2(0.665)
```

```
## [1] "substantial"
## (Rules: cohen1988)
```

Step 14: Visualize the distribution of risk categories

```
#Generate model probabilities
probs <- predict(model, testing, type = "response")
#Data frame with actual target and predicted probabilities
scores <- data.frame(
  Actual = testing$target,
  Predicted_Probability = probs
  )

# Classify the scores into risk categories
# Define risk categories based on probability thresholds
library(dplyr)
scores <- scores %>%
  mutate(
    Risk_Category = case_when(
      Predicted_Probability < 0.3 ~ "Low Risk",
      Predicted_Probability >= 0.3 & Predicted_Probability < 0.6 ~ "Medium Risk",
      Predicted_Probability >= 0.6 ~ "High Risk"
      )
    )

# Visualize the distribution of risk categories
library(ggplot2)
ggplot(scores, aes(x = Predicted_Probability, fill = Risk_Category)) +
  geom_histogram(binwidth = 0.05, color = "black") +
  labs(title = "Distribution of Predicted Probability Scores",
       x = "Predicted Probability",
       y = "Frequency")
```



```
# Review the first few rows of the score data
head(scores, 30)
```

```
##    Actual Predicted_Probability Risk_Category
## 1       0          0.1936875725      Low Risk
## 2       0          0.0934782022      Low Risk
## 3       0          0.2997032177      Low Risk
## 4       0          0.9294019274     High Risk
## 5       0          0.8761094468     High Risk
## 6       1          0.0028066440      Low Risk
## 7       0          0.1435069905      Low Risk
## 8       1          0.8063566989     High Risk
## 9       1          0.9472678160     High Risk
## 10      0          0.0036480026      Low Risk
## 11      1          0.5850682700   Medium Risk
## 12      0          0.1976715032      Low Risk
## 13      0          0.0132086376      Low Risk
## 14      1          0.9759977332     High Risk
## 15      0          0.0016043285      Low Risk
## 16      1          0.9659359017     High Risk
## 17      0          0.0667607931      Low Risk
## 18      1          0.9706073626     High Risk
## 19      0          0.0002886308      Low Risk
## 20      1          0.9930553909     High Risk
## 21      0          0.5857953139   Medium Risk
## 22      0          0.0153843213      Low Risk
## 23      1          0.9572845471     High Risk
## 24      1          0.3382015221   Medium Risk
## 25      0          0.0004665334      Low Risk
## 26      1          0.7879620087     High Risk
## 27      0          0.0228474896      Low Risk
## 28      1          0.9777087138     High Risk
## 29      1          0.5422930068   Medium Risk
## 30      0          0.0021259245      Low Risk
```

```
print(scores)
```

```
##    Actual Predicted_Probability Risk_Category
## 1       0          0.1936875725      Low Risk
## 2       0          0.0934782022      Low Risk
## 3       0          0.2997032177      Low Risk
## 4       0          0.9294019274     High Risk
## 5       0          0.8761094468     High Risk
## 6       1          0.0028066440      Low Risk
## 7       0          0.1435069905      Low Risk
## 8       1          0.8063566989     High Risk
## 9       1          0.9472678160     High Risk
## 10      0          0.0036480026      Low Risk
## 11      1          0.5850682700   Medium Risk
## 12      0          0.1976715032      Low Risk
## 13      0          0.0132086376      Low Risk
## 14      1          0.9759977332     High Risk
## 15      0          0.0016043285      Low Risk
## 16      1          0.9659359017     High Risk
## 17      0          0.0667607931      Low Risk
## 18      1          0.9706073626     High Risk
## 19      0          0.0002886308      Low Risk
## 20      1          0.9930553909     High Risk
## 21      0          0.5857953139   Medium Risk
## 22      0          0.0153843213      Low Risk
## 23      1          0.9572845471     High Risk
## 24      1          0.3382015221   Medium Risk
## 25      0          0.0004665334      Low Risk
## 26      1          0.7879620087     High Risk
## 27      0          0.0228474896      Low Risk
## 28      1          0.9777087138     High Risk
## 29      1          0.5422930068   Medium Risk
## 30      0          0.0021259245      Low Risk
## 31      1          0.7701898087     High Risk
## 32      1          0.9235837511     High Risk
## 33      1          0.2751352546      Low Risk
## 34      0          0.0003687594      Low Risk
## 35      0          0.8354342149     High Risk
## 36      1          0.9917675276     High Risk
## 37      1          0.9856179686     High Risk
## 38      1          0.9438560564     High Risk
## 39      0          0.0033757276      Low Risk
## 40      1          0.6965967541     High Risk
## 41      1          0.9072723498     High Risk
## 42      1          0.5920266141   Medium Risk
## 43      1          0.9675082453     High Risk
## 44      1          0.9992432755     High Risk
## 45      1          0.8543841244     High Risk
## 46      0          0.1999238240      Low Risk
## 47      0          0.1977395189      Low Risk
## 48      1          0.0260834106      Low Risk
## 49      0          0.3211762062   Medium Risk
## 50      1          0.7879749996     High Risk
## 51      1          0.8007829899     High Risk
## 52      1          0.9696311705     High Risk
## 53      0          0.9902360399     High Risk
## 54      0          0.0132219902      Low Risk
## 55      1          0.9614393670     High Risk
## 56      0          0.0576036172      Low Risk
## 57      1          0.0768509307      Low Risk
## 58      1          0.9854692350     High Risk
## 59      0          0.7552530243     High Risk
## 60      0          0.0527016852      Low Risk
## 61      1          0.9525372488     High Risk
```

Interpretation:

A. Confusion Matrix

1. True Positives (TP = 29): These are cases where the model correctly predicted the presence of heart disease.
2. True Negatives (TN = 21): These are cases where the model correctly predicted no heart disease.
3. False Positives (FP = 7): These are cases where the model incorrectly predicted heart disease when it wasn't present.
4. False Negatives (FN = 4): These are cases where the model failed to detect heart disease.

Key Insights: 1. The model has good predictive power, with most predictions being correct. 2. However, there are 7 false positives and 4 false negatives, which may require attention depending on the application.

B. Accuracy 1. The model correctly predicts about 82% of cases, which indicates it performs well overall. 2. However, accuracy alone may not be sufficient if the cost of false positives or false negatives is high.

C. Recall (Sensitivity) 1. The model identifies 88% of actual heart disease cases (true positives). 2. High recall is critical for medical diagnostics to minimize the number of false negatives (missed cases)

D. Precision 1. When the model predicts heart disease, it is correct 81% of the time. 2. Lower precision (compared to recall) suggests the model occasionally flags non-disease cases as heart disease (false positives)

E. F1-Score 1. The F1-Score balances Precision and Recall. 2. An F1-Score of 84% indicates the model has a good trade-off between catching most cases (high recall) and being accurate in its predictions (precision).

F. ROC Curve and AUC 1. AUC = 0.956: The model has excellent discriminatory power. 1.1 A value close to 1 means the model can reliably distinguish between patients with and without heart disease.

2. Interpretation of ROC Curve: 2.1 The curve rising steeply towards the top-left corner indicates high sensitivity and specificity for various thresholds. 2.2 Adjusting thresholds can optimize for precision or recall based on application needs

G. Risk Categorization 1. Categorizing patients into risk levels based on predicted probabilities: 2. Low Risk (< 30%): Patients unlikely to have heart disease. 3. Medium Risk (30–60%): Patients with a moderate likelihood of heart disease. 4. High Risk (> 60%): Patients with a high likelihood of heart disease.

Interpretation: 1. This segmentation helps prioritize patients for further tests or interventions. 2. For instance, “High Risk” patients might need immediate attention, while “Low Risk” patients can be monitored.

H. Key Takeaways 1. The model performs well overall, as shown by high accuracy, recall, and AUC. 2. Strength: High recall ensures most patients with heart disease are identified. 3. Limitation: Some false positives (FP) and false negatives (FN) remain, which can have implications in a clinical setting