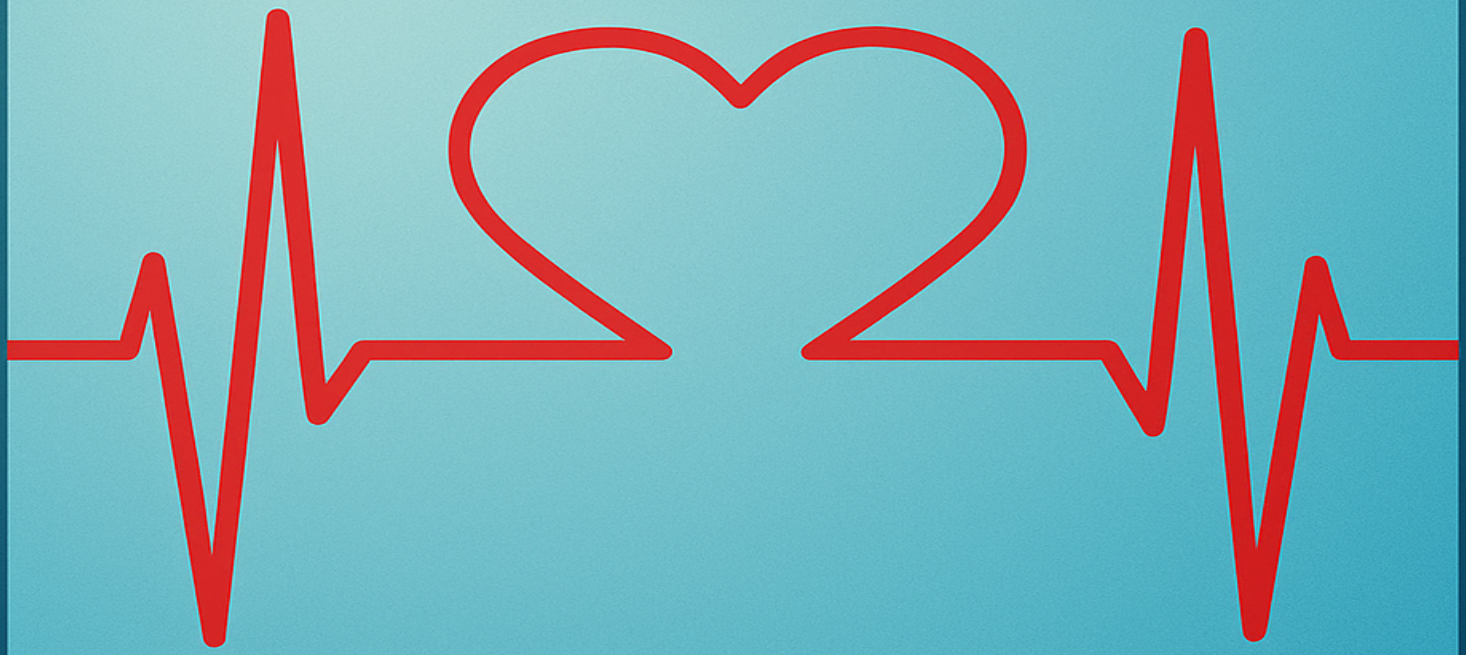
Forecasting Heart Disease Risks



Seif H. Kungulio

Forecasting Heart Disease Risks

Seif Kungulio

October 15, 2025

Contents

Business Understanding

Introduction

Cardiovascular diseases remain one of the leading causes of death worldwide, imposing significant clinical and financial burdens on individuals and organizations alike. Early identification of people at risk of developing heart disease is therefore a priority not only in healthcare but also in sectors such as insurance and public health policy. For insurance companies in particular, understanding the likelihood that a policyholder will develop heart disease is crucial for assessing health risks, pricing premiums, and designing preventive wellness programs that promote healthier lifestyles among clients.

This project applies predictive analytics and machine learning techniques to the Heart Disease dataset from the UCI Machine Learning Repository. The dataset contains a diverse range of patient attributes—including demographic, physiological, and clinical indicators—that are known to influence cardiovascular health. By systematically analyzing these variables, the project aims to uncover patterns and risk factors associated with the onset of heart disease.

The broader business objective is to build a data-driven decision support system that can forecast an individual's probability of developing heart disease. Such a model enables insurers to perform risk stratification, design personalized premium plans, and support preventive health initiatives that lower long-term costs. Through this analytical framework, the project demonstrates how data science can transform raw medical data into actionable business intelligence, fostering both economic efficiency and social impact.

Problem statement

Problem statement

To develop models for an insurance company using the Heart Disease dataset from the UCI Machine Learning Repository. The goal is to predict the likelihood of a person developing heart disease, which would help the insurance company estimate health risks and adjust premiums accordingly.

Data Understanding

The dataset contains various features related to patients' health and demographic information. We will explore the dataset to understand its structure and relationships between variables.

Data description

The Heart Disease dataset from the UCI Machine Learning Repository contains 303 instances and 14 attributes. These attributes include both numerical and categorical variables related to patients' health metrics and demographic information. The target variable indicates the presence or absence of heart disease. These attributes are:

1. `age`: Age of the patient (numeric)
2. `sex`: Gender of the patient (1 = male, 0 = female)
3. `cp`: Chest pain type (categorical: 1-4)
4. `trestbps`: Resting blood pressure (numeric)
5. `chol`: Serum cholesterol (numeric)
6. `fbs`: Fasting blood sugar (1 = true, 0 = false)
7. `restecg`: Resting electrocardiographic results (categorical)
8. `thalach`: Maximum heart rate achieved (numeric)
9. `exang`: Exercise-induced angina (1 = yes, 0 = no)
10. `oldpeak`: ST depression induced by exercise (numeric)
11. `slope`: The slope of the peak exercise ST segment (categorical)
12. `ca`: Number of major vessels (0-3, numeric)
13. `thal`: Thalassemia (categorical: 1 = normal, 2 = fixed defect, 3 = reversible defect)
14. `target`: Heart disease (1 = disease, 0 = no disease)

Data dictionary

The dataset contains 14 key attributes that are either numerical or categorical.

Attribute	Type	Description	Constraints/ Rules
`age`	Numerical	The age of the patient in years	Range: 29-77 (based on dataset statistics)
`sex`	Categorical	The gender of the patient	Values: 1 = Male, 0 = Female
`cp`	Categorical	Type of chest pain experienced by the patient	Values: 1 = Typical angina, 2 = Atypical angina, 3 = Non-anginal pain, 4 = Asymptomatic
`trestbps`	Numerical	Resting blood pressure of the patient, measured in mmHg	Range: Typically, between 94 and 200 mmHg
`chol`	Numerical	Serum cholesterol level in mg/dl	Range: Typically, between 126 and 564 mg/dl
`fbs`	Categorical	Fasting blood sugar level > 120 mg/dl	Values: 1 = True, 0 = False
`restecg`	Categorical	Results of the patient's resting electrocardiogram	Values: 0 = Normal, 1 = ST-T wave abnormality, 2 = Probable or definite left ventricular hypertrophy

Attribute	Type	Description	Constraints/ Rules
`thalach`	Numerical	Maximum heart rate achieved during a stress test	Range: Typically, between 71 and 202 bpm
`exang`	Categorical	Whether the patient experiences exercise-induced angina	Values: 1 = Yes, 0 = No
`oldpeak`	Numerical	ST depression induced by exercise relative to rest (an ECG measure)	Range: 0.0 to 6.2 (higher values indicate more severe abnormalities)
`slope`	Categorical	Slope of the peak exercise ST segment	Values: 1 = Upsloping, 2 = Flat, 3 = Downsloping
`ca`	Numerical	Number of major vessels colored by fluoroscopy	Range: 0-3
`thal`	Categorical	Blood disorder variable related to thalassemia	Values: 3 = Normal, 6 = Fixed defect, 7 = Reversible defect
`target`	Categorical	Diagnosis of heart disease	Values: 0 = No heart disease, 1 = Presence of heart disease

Initial observations

- The dataset contains a mix of numerical and categorical variables.
- Some variables may require preprocessing, such as handling missing values and encoding categorical variables.
- Missing Values: Some fields like ca and thal may have missing values or unknown entries ('?').
- Data Types: Some categorical variables are encoded numerically and will need to be interpreted correctly during analysis.
- Class Imbalance: Preliminary checks suggest the dataset is relatively balanced between presence and absence of disease, but this will be verified.
- Outliers: Numerical fields such as chol (cholesterol) and trestbps (blood pressure) may have outliers that need to be detected and considered in analysis.

Data Preparation

Data loading

Load the dataset from the UCI website to memory

```
Heart.df <- read.csv(text = getURL(url), header = FALSE, na.strings = "?")
```

When I loaded the dataset using the `read.csv()` function, I made sure to convert the placeholder `"?"` values into proper `NA` entries. This was important because the original UCI file embeds missing values as question marks, which R would otherwise treat as literal characters. By handling this upfront, I ensured that all subsequent preprocessing steps—such as imputation, transformations, and statistical summaries—would correctly recognize missing data. This step was the foundation that allowed me to work with a clean and interpretable dataset from the very beginning.

Display dimensions of the dataset

```
dim(Heart.df)
```

```
## [1] 303  14
```

After loading the data, I checked the dimensions and confirmed that the dataset contained **303 records and 14 variables**. For me, this was a critical validation step because it reassured me that the dataset had been imported completely, without truncation or corruption. Knowing the exact dimensions also helped me plan the appropriate modeling strategy. With 303 observations, I recognized that I needed to be cautious about overfitting and choose models that perform well on modest-sized datasets.

Display the first six rows of the dataset

```
head(Heart.df)
```

```
##   V1 V2 V3  V4  V5 V6 V7  V8 V9 V10 V11 V12 V13 V14
## 1 63  1  1 145 233  1  2 150  0 2.3   3   0   6   0
## 2 67  1  4 160 286  0  2 108  1 1.5   2   3   3   2
## 3 67  1  4 120 229  0  2 129  1 2.6   2   2   7   1
## 4 37  1  3 130 250  0  0 187  0 3.5   3   0   3   0
## 5 41  0  2 130 204  0  2 172  0 1.4   1   0   3   0
## 6 56  1  2 120 236  0  0 178  0 0.8   1   0   3   0
```

Inspecting the first few rows of the dataset gave me an initial sense of its structure in raw form. The variables were still labeled generically as `V1` through `V14`, and I could already see that categorical fields—such as `ca` and `thal`—contained `"?"` values. This preview confirmed that data quality issues were present and would require careful cleaning. It also highlighted that the dataset treated all variables as numeric values, even when some of them clearly represented categories. Seeing these patterns early helped me anticipate the transformations and encodings that would be needed later.

Rename the columns into a meaningful column names

```
colnames(Heart.df) <- c("age", "sex", "cp", "trestbps", "chol", "fbs", "restecg",
                        "thalach", "exang", "oldpeak", "slope", "ca", "thal", "target")
```

Renaming the default column headers to clinically meaningful names was an important step for both clarity and accuracy. By assigning labels such as `age`, `trestbps`, `chol`, and `thalach`, I transformed an ambiguous numeric matrix into a dataset with clearly defined medical variables. This made my analysis much more intuitive and significantly reduced the risk of errors during visualization or modeling. From this point forward, every plot, summary, and model became easier to interpret in a clinical and analytical context.

Display the structure of the dataframe

```
glimpse(Heart.df)
```

```
## Rows: 303
## Columns: 14
## $ age      <dbl> 63, 67, 67, 37, 41, 56, 62, 57, 63, 53, 57, 56, 56, 44, 52, 5~
## $ sex      <dbl> 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1~
## $ cp       <dbl> 1, 4, 4, 3, 2, 2, 4, 4, 4, 4, 4, 2, 3, 2, 3, 3, 2, 4, 3, 2, 1~
## $ trestbps <dbl> 145, 160, 120, 130, 130, 120, 140, 120, 130, 140, 140, 140, 1~
## $ chol     <dbl> 233, 286, 229, 250, 204, 236, 268, 354, 254, 203, 192, 294, 2~
## $ fbs      <dbl> 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0~
## $ restecg  <dbl> 2, 2, 2, 0, 2, 0, 2, 0, 2, 2, 0, 2, 2, 0, 0, 0, 0, 0, 0, 0, 2~
## $ thalach  <dbl> 150, 108, 129, 187, 172, 178, 160, 163, 147, 155, 148, 153, 1~
## $ exang    <dbl> 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1~
## $ oldpeak  <dbl> 2.3, 1.5, 2.6, 3.5, 1.4, 0.8, 3.6, 0.6, 1.4, 3.1, 0.4, 1.3, 0~
## $ slope    <dbl> 3, 2, 2, 3, 1, 1, 3, 1, 2, 3, 2, 2, 2, 1, 1, 1, 3, 1, 1, 1, 2~
## $ ca       <dbl> 0, 3, 2, 0, 0, 0, 2, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0~
## $ thal     <dbl> 6, 3, 7, 3, 3, 3, 3, 3, 7, 7, 6, 3, 6, 7, 7, 3, 7, 3, 3, 3, 3~
## $ target   <int> 0, 2, 1, 0, 0, 0, 3, 0, 2, 1, 0, 0, 2, 0, 0, 0, 1, 0, 0, 0, 0~
```

Using `glimpse()`, I inspected the internal structure of the dataset and confirmed that all variables were loaded as numeric types. Although this is typical for UCI data, it reinforced the need to convert several columns—such as `sex`, `cp`, `exang`, `restecg`, and `thal`—into factors. Additionally, I noticed that the `target` variable contained values beyond just 0 and 1. This told me that the dataset initially encoded heart disease severity on a scale rather than a simple binary indicator. Recognizing this early helped me plan the binary conversion that would later make the dataset suitable for classification models.

Display the statistical summary of the dataframe

```
summary(Heart.df)
```

```
##       age             sex               cp           trestbps
##  Min.   :29.00   Min.   :0.0000   Min.   :1.000   Min.   : 94.0
##  1st Qu.:48.00   1st Qu.:0.0000   1st Qu.:3.000   1st Qu.:120.0
##  Median :56.00   Median :1.0000   Median :3.000   Median :130.0
##  Mean   :54.44   Mean   :0.6799   Mean   :3.158   Mean   :131.7
##  3rd Qu.:61.00   3rd Qu.:1.0000   3rd Qu.:4.000   3rd Qu.:140.0
##  Max.   :77.00   Max.   :1.0000   Max.   :4.000   Max.   :200.0
##
##       chol            fbs            restecg          thalach
##  Min.   :126.0   Min.   :0.0000   Min.   :0.0000   Min.   : 71.0
##  1st Qu.:211.0   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:133.5
##  Median :241.0   Median :0.0000   Median :1.0000   Median :153.0
##  Mean   :246.7   Mean   :0.1485   Mean   :0.9901   Mean   :149.6
##  3rd Qu.:275.0   3rd Qu.:0.0000   3rd Qu.:2.0000   3rd Qu.:166.0
##  Max.   :564.0   Max.   :1.0000   Max.   :2.0000   Max.   :202.0
##
```

```
##      exang           oldpeak         slope             ca
##  Min.   :0.0000   Min.   :0.00   Min.   :1.000   Min.   :0.0000
##  1st Qu.:0.0000   1st Qu.:0.00   1st Qu.:1.000   1st Qu.:0.0000
##  Median :0.0000   Median :0.80   Median :2.000   Median :0.0000
##  Mean   :0.3267   Mean   :1.04   Mean   :1.601   Mean   :0.6722
##  3rd Qu.:1.0000   3rd Qu.:1.60   3rd Qu.:2.000   3rd Qu.:1.0000
##  Max.   :1.0000   Max.   :6.20   Max.   :3.000   Max.   :3.0000
##                                                  NA's   :4
##       thal           target
##  Min.   :3.000   Min.   :0.0000
##  1st Qu.:3.000   1st Qu.:0.0000
##  Median :3.000   Median :0.0000
##  Mean   :4.734   Mean   :0.9373
##  3rd Qu.:7.000   3rd Qu.:2.0000
##  Max.   :7.000   Max.   :4.0000
##  NA's   :2
```

The summary statistics provided the first quantitative snapshot of the dataset. For instance, I observed that age ranged from 29 to 77, cholesterol included extremely high values reaching 564 mg/dl, and the `ca` and `thal` variables contained missing entries. The summaries also revealed skewness in variables like `oldpeak`, which aligns with the clinical behavior of ST depression measurements. Seeing these descriptive statistics helped me identify potential outliers and underscored the importance of filtering and standardizing certain variables. It also confirmed that the target variable included multiple values, further reinforcing the need for recoding before building predictive models.

Data preprocessing

We will preprocess the data by handling missing values, encoding categorical variables, and scaling numerical features.

Convert binary variables

According to the data dictionary, the following attributes should be binary variables: `sex`, `fbs`, `exang`, and `target`. But, some shows to have values besides 0's and 1's. Let's convert binary variables to (0, 1)

```
Heart.df$target <- ifelse(Heart.df$target > 0, 1, 0)
Heart.df$sex <- ifelse(Heart.df$sex > 0, 1, 0)
Heart.df$fbs <- ifelse(Heart.df$fbs > 0, 1, 0)
Heart.df$exang <- ifelse(Heart.df$exang > 0, 1, 0)
```

In the first stage of preprocessing, I converted several variables—target, sex, fbs, and exang—into consistent 0/1 indicators. For target, this step collapses the original multi-level heart disease severity scale into a straightforward binary outcome, where 1 represents any level of diagnosed disease and 0 represents no disease. For the remaining variables, this transformation ensures that each behaves as a clean binary predictor, free from stray numeric values that may have existed in the raw dataset. This not only standardizes their structure but also simplifies interpretation during modeling, particularly for classification algorithms that rely on clearly defined factor boundaries.

Handle missing values

Handle missing values in `ca` and `thal` variables using median imputation.

```
Heart.df$ca[is.na(Heart.df$ca)] <- median(Heart.df$ca, na.rm = TRUE)
Heart.df$thal[is.na(Heart.df$thal)] <- median(Heart.df$thal, na.rm = TRUE)
```

Next, I addressed missing data in the `ca` and `thal` variables by replacing their NA values with the respective medians. I selected median imputation because these variables tend to be skewed in clinical datasets, and the median provides a robust estimate that is less influenced by extreme values. By doing so, I retained all patient records rather than removing rows with incomplete values, which is essential given the relatively small size of the Cleveland dataset. This decision preserves data integrity while ensuring that downstream models receive complete and usable inputs.

Check for missing values if still exist

```
sapply(Heart.df, function(x) sum(is.na(x)))
```

```
##      age      sex       cp trestbps     chol      fbs  restecg  thalach
##        0        0        0        0        0        0        0        0
##    exang  oldpeak    slope       ca     thal   target
##        0        0        0        0        0        0
```

After imputation, I performed a dataset-wide check to ensure that no missing values remained. By applying a simple column-wise NA count, I confirmed that every variable now contains zero missing entries. This validation step reassures me that the dataset is ready for modeling techniques that require fully complete inputs, and it eliminates the need for additional imputation layers during the modeling workflow. Having a dataset with no missing values also improves the reliability of exploratory visualizations and summary statistics.

Handle duplicate entries

Check for duplicate entries and print them if they exist.

```
dupes <- Heart.df[duplicated(Heart.df) | duplicated(Heart.df, fromLast = TRUE), ]
print(dupes)
```

```
##  [1] age      sex      cp       trestbps chol     fbs      restecg  thalach
##  [9] exang    oldpeak  slope    ca       thal     target
## <0 rows> (or 0-length row.names)
```

I also checked for duplicated rows to ensure that each patient is represented only once. Using both forward and backward duplication checks allowed me to identify any fully repeated rows in the dataset. The results showed that no duplicates were present. This verification step is important because duplicated observations could bias model estimates, especially in a medical context where repeated entries could artificially strengthen relationships or inflate prevalence rates. With this confirmation, I can proceed confidently knowing that every record contributes unique information.

Convert categorical variables

Define a list of categorical columns with their levels and labels

```r
categorical_columns <- list(
  sex = list(levels = c(0, 1),
             labels = c("Female", "Male")),
  cp = list(levels = c(1, 2, 3, 4),
            labels = c("Typical Angina", "Atypical Angina", "Non-Angina",
                       "Asymptomatic")),
  fbs = list(levels = c(0, 1),
             labels = c("False", "True")),
  restecg = list(levels = c(0, 1, 2),
                 labels = c("Normal", "Wave-abnormality", "Probable")),
  exang = list(levels = c(0, 1),
               labels = c("No", "Yes")),
  slope = list(levels = c(1, 2, 3),
               labels = c("Upsloping", "Flat", "Downsloping")),
  thal = list(levels = c(3, 6, 7),
              labels = c("Normal", "Fixed Defect", "Reversible")),
  target = list(levels = c(1, 0),
                labels = c("Yes", "No"))
)
```

To improve interpretability, I created a structured mapping that assigns descriptive labels to the categorical variables such as chest pain type (cp), resting ECG results (restecg), exercise-induced angina (exang), and thalassemia status (thal). By translating raw numeric codes into clinically meaningful terms—for example, converting chest pain type "4" into "Asymptomatic"—I ensured that all future summaries, model outputs, and visualizations reflect information that is easy to understand. This labeling not only improves readability but also reduces the likelihood of misinterpretation when presenting findings to non-technical stakeholders.

Apply the factor transformation using a for-loop.

```r
for (col in names(categorical_columns)) {
  Heart.df[[col]] <- factor(Heart.df[[col]],
                            levels = categorical_columns[[col]]$levels,
                            labels = categorical_columns[[col]]$labels)
}
```

With the label mappings in place, I programmatically converted each categorical variable into a factor using a loop. This ensures that R treats these fields appropriately during modeling and visualization. Establishing explicit levels and labels is especially valuable for classification tasks, effect plots, and bar charts, where the ordering of categories can influence interpretation. This systematic conversion step brings consistency across variables and prepares the dataset for tidyverse and tidymodel workflows.

Handle outliers in numerical variables

Apply multiple filters to identify and handle outliers in numerical variables.

```r
Heart.df <- Heart.df[Heart.df$age > 40 &
                       Heart.df$trestbps < 170 &
                       Heart.df$chol < 340 &
                       Heart.df$chol > 150 &
                       Heart.df$thalach > 115 &
                       Heart.df$oldpeak < 2.4, ]
```

This section of the analysis performs a crucial data-cleaning step aimed at refining the quality of the dataset before modeling and visualization. The filtering operation applies a set of logical conditions to remove extreme or biologically implausible values from key continuous health indicators such as age, blood pressure, cholesterol, heart rate, and ST depression. By doing so, it ensures that the dataset reflects realistic patient characteristics and minimizes the influence of outliers that could distort statistical interpretation or predictive accuracy.

The first filter, `Heart.df$age > 40`, narrows the focus to patients over 40 years of age. This decision is grounded in clinical reasoning—heart disease is relatively uncommon in younger individuals, and including them could introduce noise rather than insight into cardiovascular risk patterns. The next condition, `Heart.df$trestbps < 170`, restricts resting blood pressure to physiologically typical values, removing excessively high readings that may result from measurement error or rare hypertensive crises.

Similarly, cholesterol values are filtered using two constraints: `Heart.df$chol < 340` and `Heart.df$chol > 150`. This dual boundary ensures that cholesterol readings fall within a realistic clinical range, excluding both unusually low and excessively high values. Extremely high cholesterol levels (above 340 mg/dl) could be outliers due to lab errors or rare genetic conditions, while very low levels (below 150 mg/dl) are equally atypical for this patient population.

The condition `Heart.df$thalach > 115` retains only those patients whose maximum heart rate achieved during exercise falls within a normal performance range. Extremely low thalach values often suggest incomplete stress tests or data entry errors, which could bias the interpretation of cardiovascular efficiency. Finally, `Heart.df$oldpeak < 2.4` removes extreme ST depression values. In clinical terms, oldpeak measures the degree of ST segment depression during exercise, and values beyond 2.4 are uncommon and may represent atypical cardiac events that do not align with general population patterns in the dataset.

Overall, these filters collectively enhance the integrity of the data and the reliability of subsequent analysis. By trimming implausible extremes, the dataset becomes more homogeneous, improving the clarity of boxplots, histograms, and scatterplots generated during exploratory data analysis. Moreover, this targeted filtering supports more stable and interpretable model outcomes by preventing a few extreme observations from disproportionately influencing trends or coefficients. The result is a dataset that better represents realistic health profiles, ultimately strengthening the credibility of insights drawn from the heart disease risk modeling process.

Helper functions

Function to create Box plots

```
HeartDiseaseBoxplot <- function(var1, var2) {
  ggplot(Heart.df, aes(x = .data[[var1]],
                       y = .data[[var2]],
                       fill = .data[[var1]])) +
    geom_boxplot() +
    labs(title = paste("Boxplot of", var2, "by", var1),
         x = var1, y = var2, fill = "Heart Disease")
}
```

The HeartDiseaseBoxplot() function is designed to visualize how a numerical health-related variable behaves across the different levels of a categorical grouping variable, typically the heart disease outcome. By placing the grouping variable on the x-axis and a continuous risk factor on the y-axis, the resulting boxplot highlights differences in median values, variability, and outliers for each group. This plot offers a fast and intuitive way to assess how a particular clinical measurement—such as age, blood pressure, cholesterol, maximum heart rate, or ST depression—differs between patients with and without diagnosed heart disease. It plays a critical role in early exploratory analysis, helping uncover potential risk factor patterns that may influence model development later.

Function to create Bar plots

```
HeartDiseaseBar <- function(var) {
  ggplot(Heart.df, aes(x = .data[[var]], fill = target)) +
    geom_bar(position = "dodge") +
    labs(title = paste("Distribution of Heart Disease by", var),
         x = var, fill = "Heart Disease")
}
```

The HeartDiseaseBar() function produces a categorical bar chart that compares the counts of heart disease outcomes within each category of a selected variable. This is especially useful for demographic or clinical classification variables such as sex, chest pain type, fasting blood sugar, resting ECG results, exercise-induced angina, slope, or thalassemia status. By breaking down the target variable across these groups, the bar chart reveals which categories show higher or lower observed heart disease prevalence. This insight is valuable in understanding population segments that may require additional attention, and it helps identify categorical predictors that could contribute significantly to a classification model.

Function to create Histograms

```
HeartDiseaseHist <- function(var1) {
  ggplot(Heart.df, aes(x = .data[[var1]], fill = target)) +
    geom_histogram(bins = 15) +
    labs(title = paste("Distribution of", var1),
         x = var1, fill = "Heart Disease")
}
```

The HeartDiseaseHist() function creates a histogram for a continuous health variable and overlays the distribution according to heart disease status. This visualization makes it easy to examine the overall shape, skewness, and concentration of values for a risk factor while simultaneously comparing the distribution between patients with and without heart disease. Whether analyzing age, resting blood pressure, cholesterol levels, maximum heart rate achieved, or ST depression, the histogram reveals whether certain ranges of values appear more frequently in either outcome group. This function enhances exploratory data analysis by showing how numerical features behave across the target variable and whether separation patterns exist that the model may later exploit.

Function to create Scatter plots

```
HeartDiseaseScatter <- function(point1, point2){
  ggplot(Heart.df, aes(x = .data[[point1]],
                       y = .data[[point2]],
                       color = target)) +
    geom_point(size = 2) +
    geom_smooth(method = "lm", se = FALSE, color = "blue", formula = y ~ x) +
    labs(title = paste("Scatterplot of", point1, "by", point2),
       x = point1, y = point2, color = "Heart Disease")
}
```

The HeartDiseaseScatter() function generates a scatterplot showing the relationship between two numeric variables, with each point colored by heart disease status. By plotting one clinical indicator on the x-axis and

another on the y-axis, this visualization helps reveal correlations, clusters, and potential separation between outcome groups. The function also overlays a linear trend line for each group, giving a quick visual summary of the direction and strength of the relationship. This makes the scatterplot especially useful for exploring associations such as age versus cholesterol, heart rate versus ST depression, or blood pressure versus heart rate. Ultimately, this function supports deeper understanding of how pairs of predictors interact and how those interactions relate to the presence of heart disease.

Exploratory data analysis

Class imbalance

```
Heart.df %>% count(target) %>% mutate(pct = n/sum(n))
```

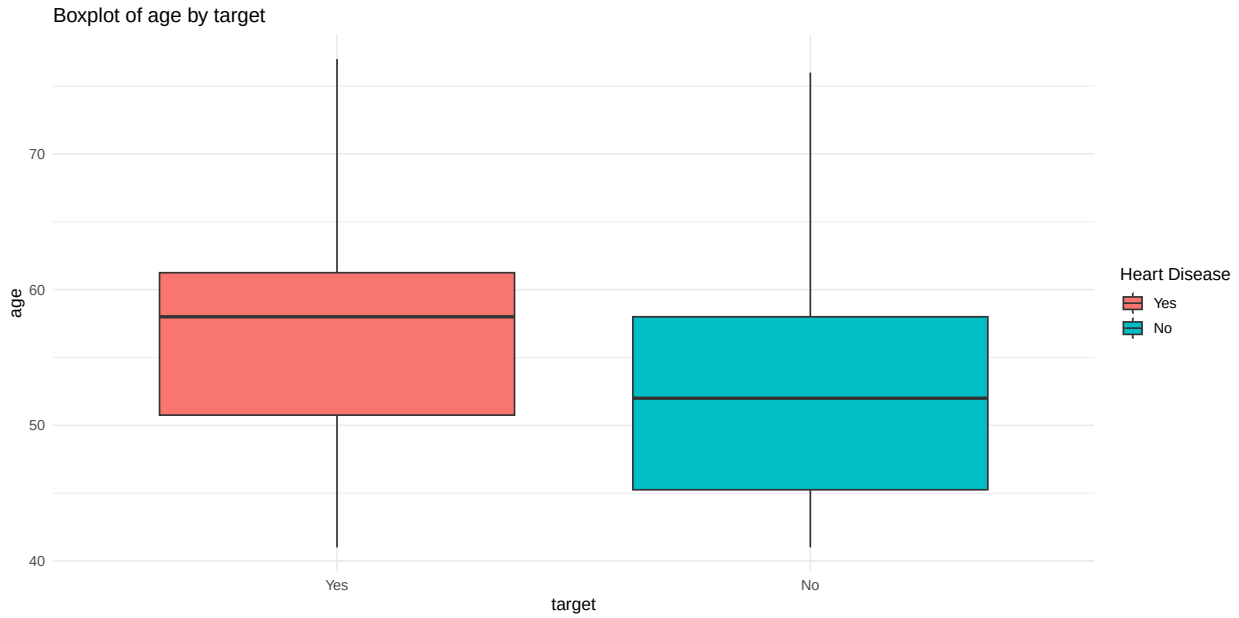
```
##   target   n       pct
## 1    Yes  68 0.3434343
## 2     No 130 0.6565657
```

Boxplots for numerical variables

I used boxplots to visually examine the distribution of key continuous health indicators — such as age, resting blood pressure (trestbps), cholesterol (chol), maximum heart rate (thalach), and ST depression (oldpeak) — across the binary target variable (Heart Disease: Yes / No). Boxplots were chosen because they efficiently highlight differences in central tendency (median), variability (IQR), and the presence of potential outliers between patients with and without heart disease.

Boxplot of Age by Heart Disease

```
HeartDiseaseBoxplot("target", "age")
```

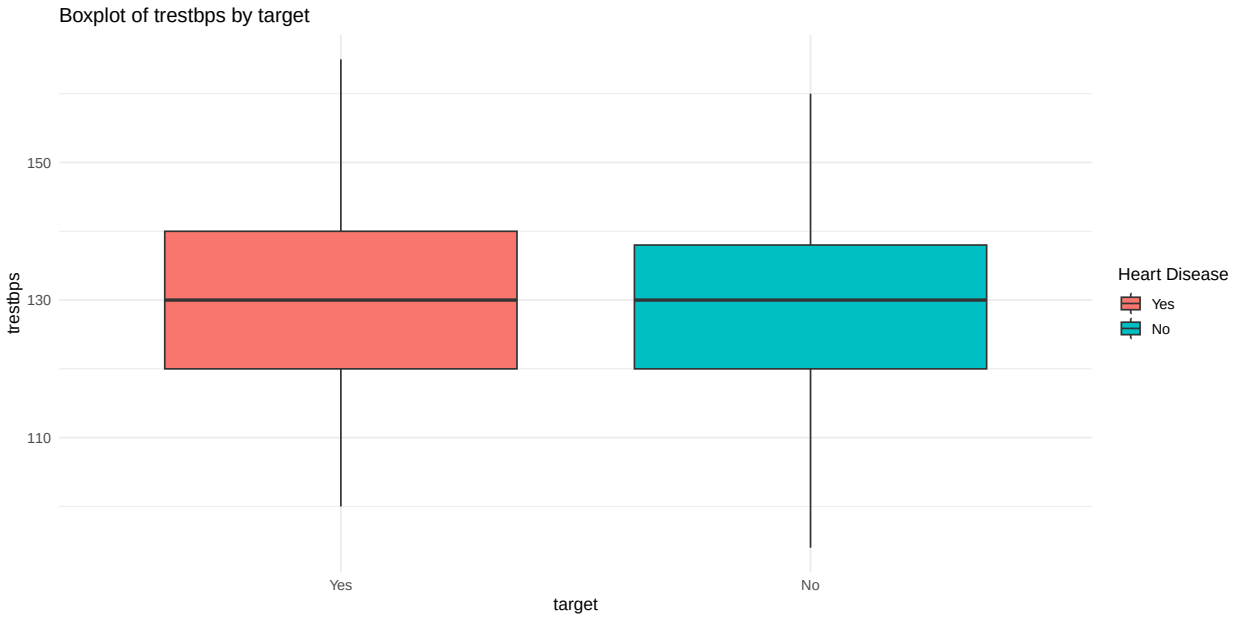


The age boxplot shows that both groups are relatively similar in age distribution, but individuals with heart disease tend to be slightly older on average. The median age for the “Yes” group appears marginally

higher, suggesting that age remains a contributing factor even after filtering out younger patients below 40. However, the overlap in interquartile ranges indicates that age alone does not offer strong separation between the two groups.

Boxplot of Resting Blood Pressure (trestbps) by Heart Disease

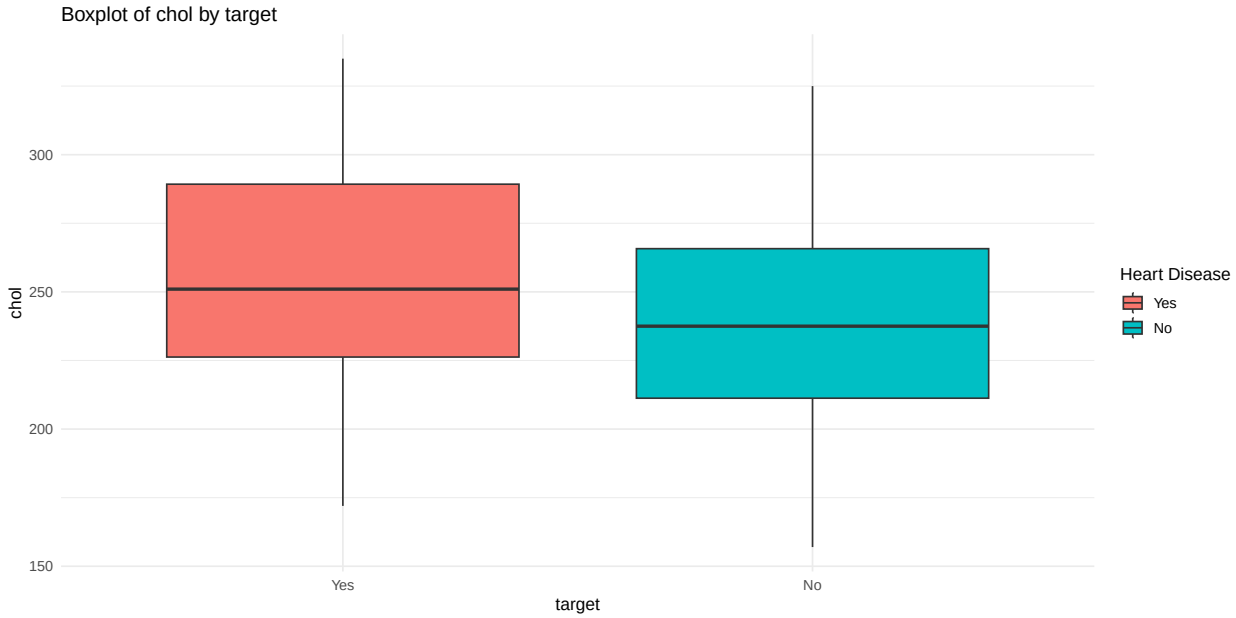
```
HeartDiseaseBoxplot("target", "trestbps")
```



The trestbps boxplot indicates that patients with heart disease have resting blood pressure levels that are generally comparable to those without heart disease. The medians of both groups are close, and their spreads overlap significantly. This suggests that resting blood pressure—after extreme values were filtered—does not strongly differentiate between the two groups.

Boxplot of Cholesterol (chol) by Heart Disease

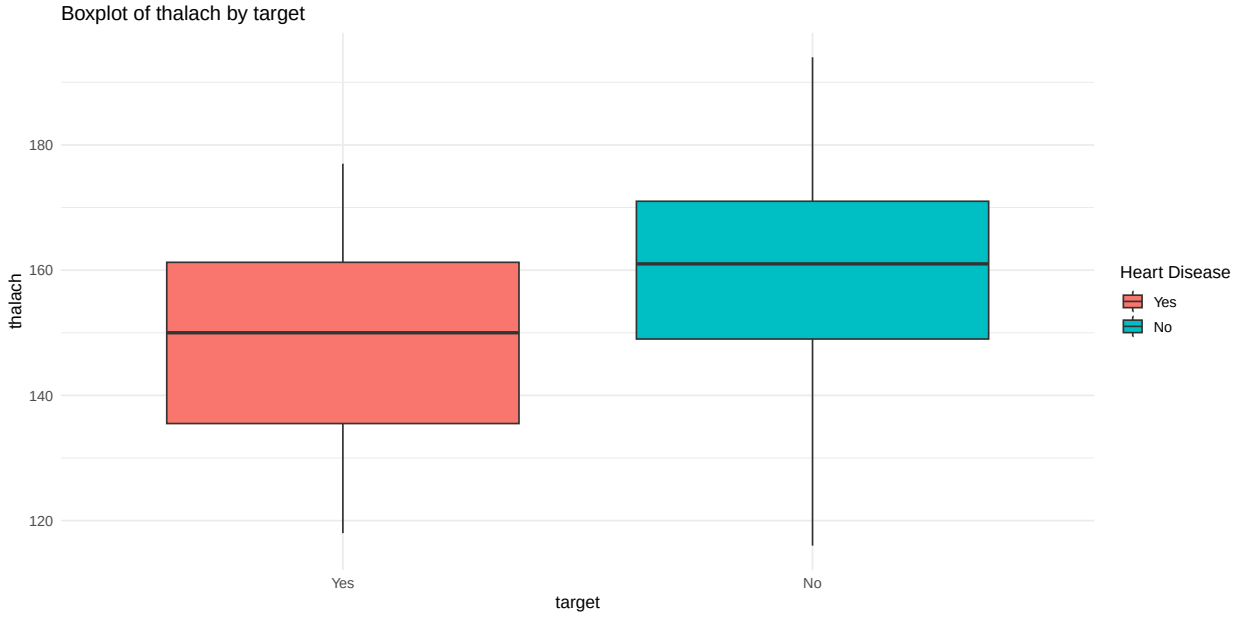
```
HeartDiseaseBoxplot("target", "chol")
```



The cholesterol boxplot shows slightly higher cholesterol values among the heart-disease group, with a marginally higher median level. Although there is considerable overlap in distributions, the "Yes" group tends to include more values toward the upper range. This pattern is consistent with cholesterol being a known cardiovascular risk factor, but not a deterministic one within this dataset.

Boxplot of Maximum Heart Rate Achieved (thalach) by Heart Disease

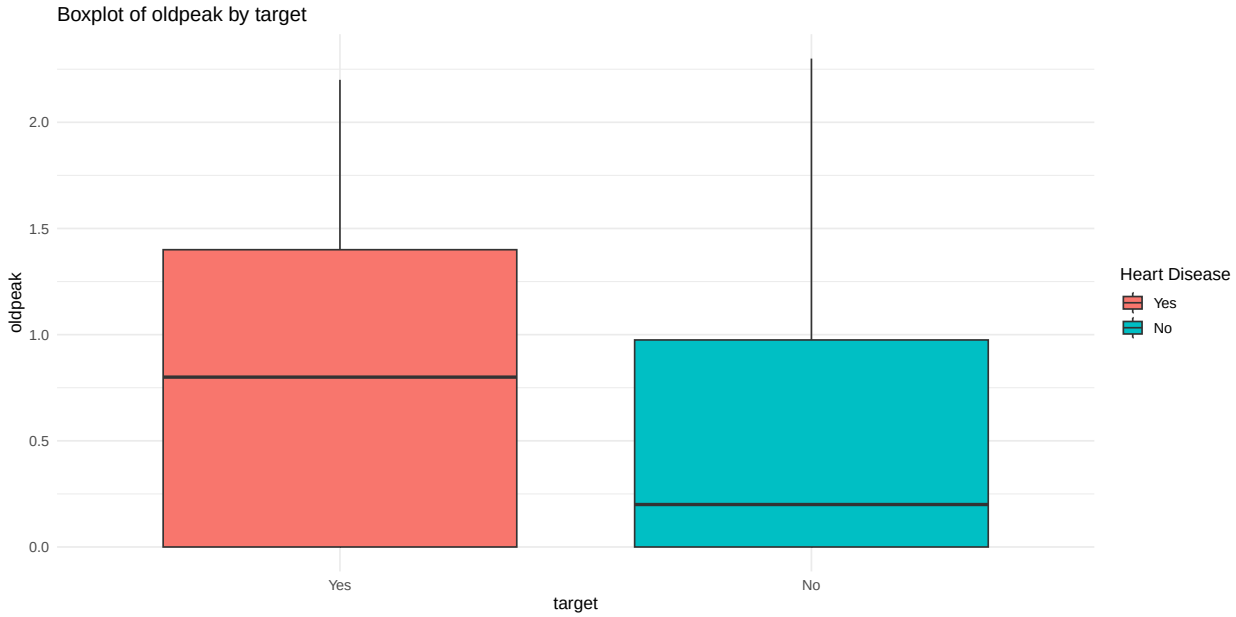
```
HeartDiseaseBoxplot("target", "thalach")
```



The thalach boxplot reveals a clearer distinction than the previous variables: patients with heart disease tend to have lower maximum heart rates achieved during exercise. The median thalach for the "Yes" group is visibly lower, and their distribution is shifted downward relative to the "No" group. This aligns with clinical expectations—reduced cardiac performance often manifests as lower achievable heart rates during stress tests.

Boxplot of ST Depression (oldpeak) by Heart Disease

```
HeartDiseaseBoxplot("target", "oldpeak")
```



This boxplot provides one of the strongest visual separations. Patients with heart disease generally exhibit higher oldpeak values, indicating greater ST depression during exercise. The median for the "Yes" group is noticeably higher, and the upper range of ST depression appears more pronounced. Because ST depression is directly linked to abnormal myocardial response, this variable stands out as a meaningful predictor.

Overall boxplots observations:

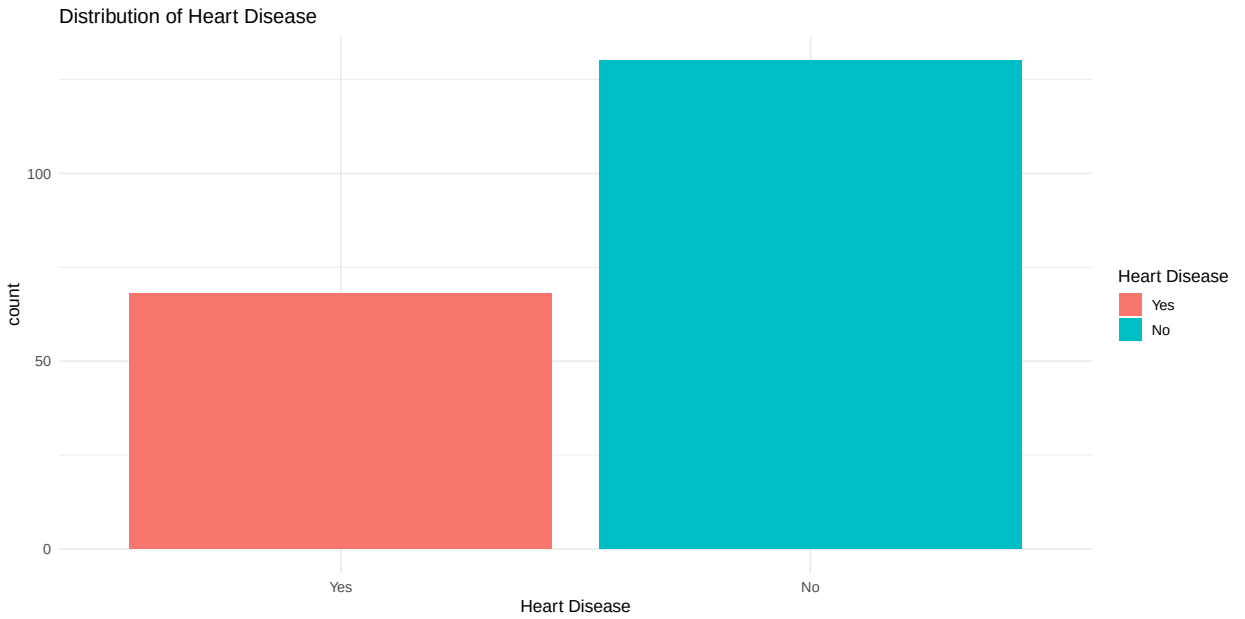
Across all numerical variables, thalach and oldpeak show the most distinct separation between heart-disease and non-heart-disease groups. Age and cholesterol exhibit moderate differentiation, while resting blood pressure shows minimal separation.

These insights support the modeling phase by identifying which numeric predictors carry the strongest signal for classification.

Barplots for categorical variables

Heart disease distribution

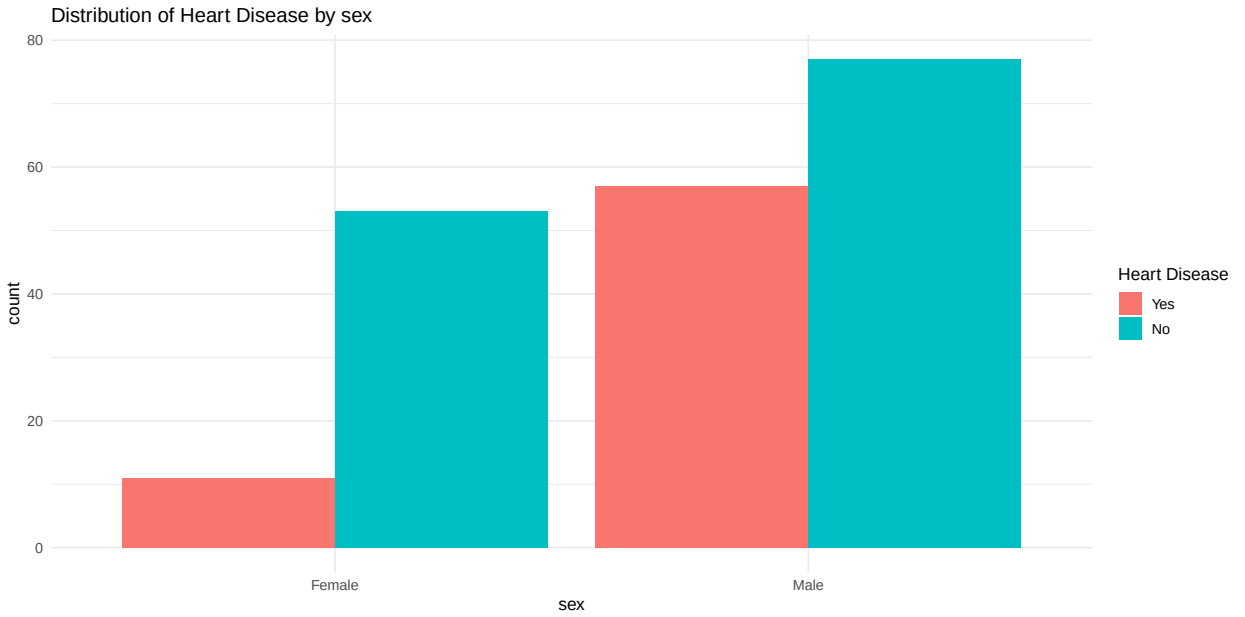
```
ggplot(Heart.df, aes(x=target, fill=target))+
  geom_bar() +
  ggtitle("Distribution of Heart Disease") +
  labs(x = "Heart Disease", fill = "Heart Disease")
```



The barplot of the target variable shows that the dataset contains more patients without heart disease than those diagnosed with the condition. While the groups are not perfectly balanced, the distribution is still sufficient to support reliable classification modeling. The presence of a substantial number of patients in both groups ensures that patterns distinguishing diseased from non-diseased individuals can be learned without the need for aggressive resampling techniques.

Visualize distribution of sex

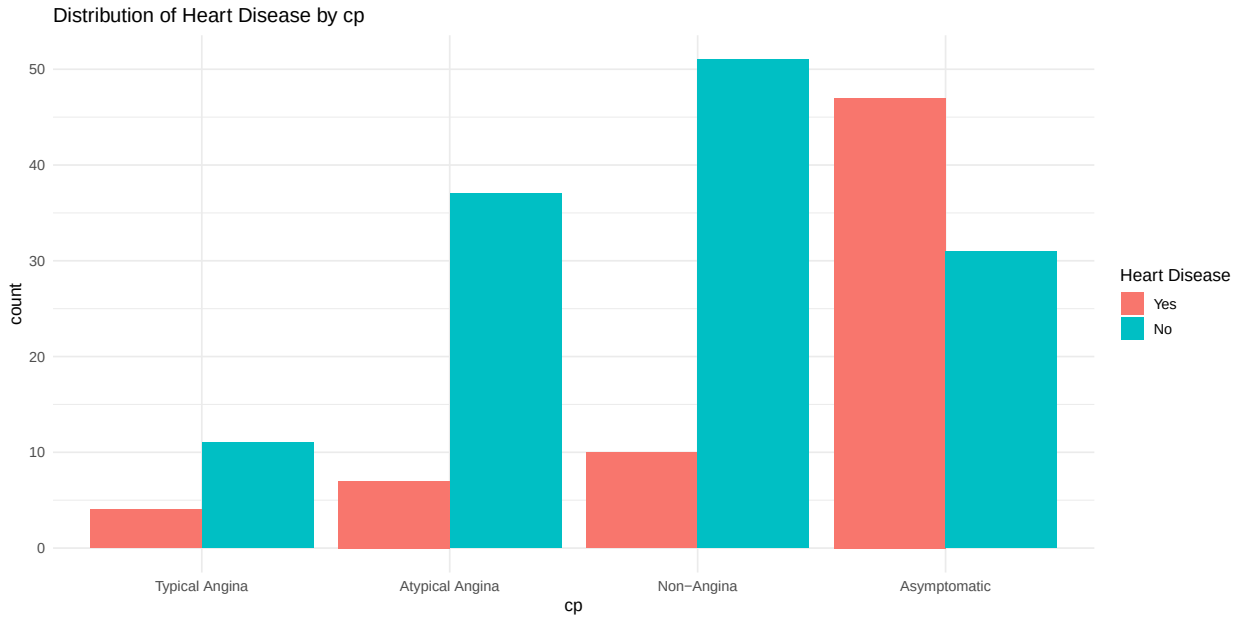
```
HeartDiseaseBar("sex")
```



The distribution of heart disease across sex reveals a noticeable difference between males and females. Although both groups include patients with and without heart disease, the proportion of heart disease cases is higher among males. This observation aligns with established clinical evidence showing that men are at a higher risk for developing cardiovascular conditions. The barplot therefore suggests that sex may hold moderate predictive value in distinguishing risk levels.

Visualize distribution of `cp`

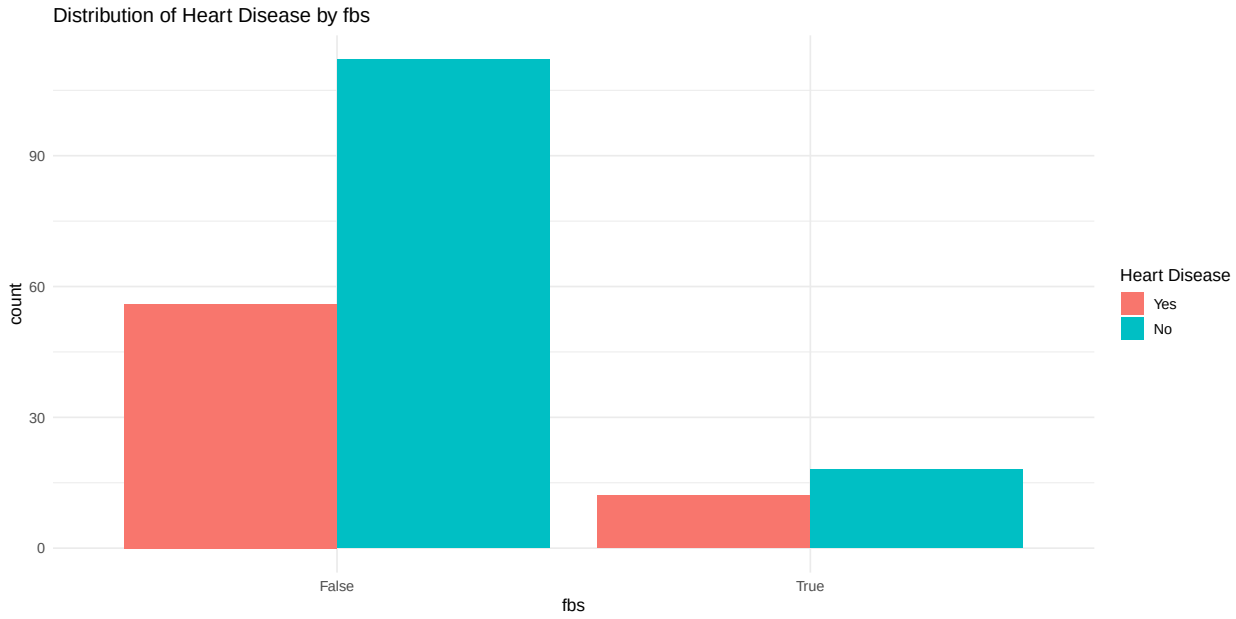
```
HeartDiseaseBar("cp")
```



Chest pain type stands out as one of the most informative categorical variables in the dataset. The barplot shows that patients with asymptomatic chest pain (classified as type 4) have a markedly higher prevalence of heart disease compared to the other chest pain categories. In contrast, typical and atypical angina types show fewer heart disease cases, and non-anginal discomfort falls somewhere in between. This strong separation highlights chest pain type as a critical clinical marker and a powerful predictor for modeling.

Visualize distribution of `fbs`

```
HeartDiseaseBar("fbs")
```

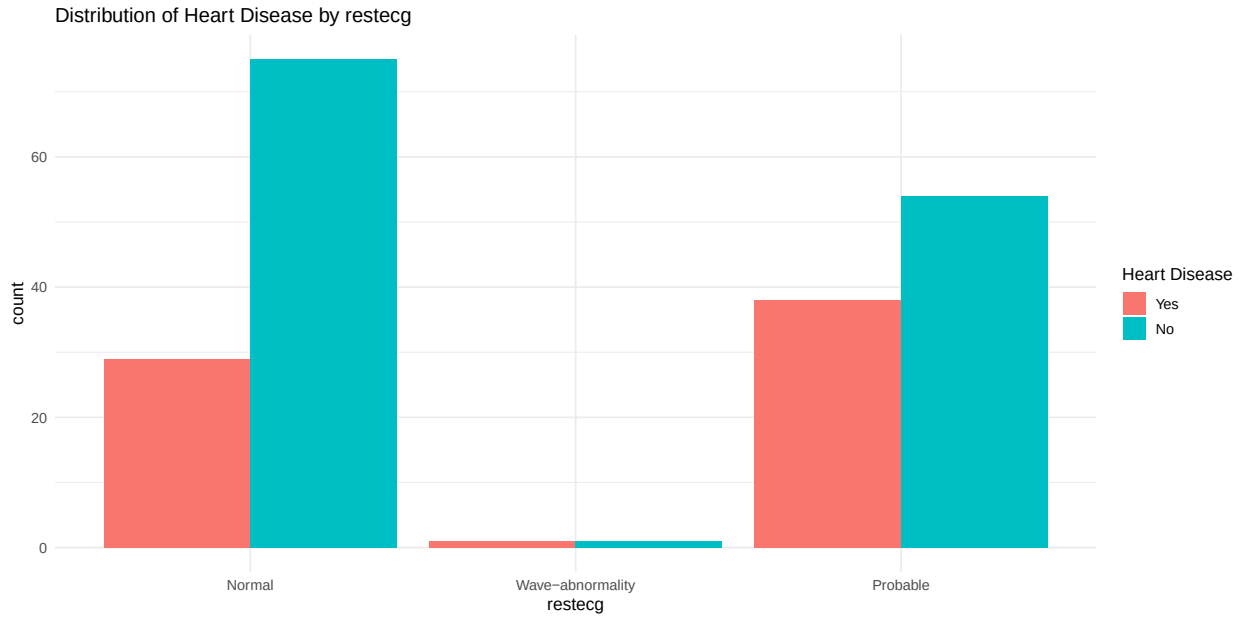


The comparison between patients with elevated fasting blood sugar and those within normal ranges reveals

minimal separation in heart disease outcomes. Both categories show similar proportions of "Yes" and "No" cases, suggesting that fasting blood sugar greater than 120 mg/dl does not substantially influence heart disease prevalence in this particular dataset. As a result, fbs appears to contribute only weak predictive value to the classification task.

Visualize distribution of `restecg`

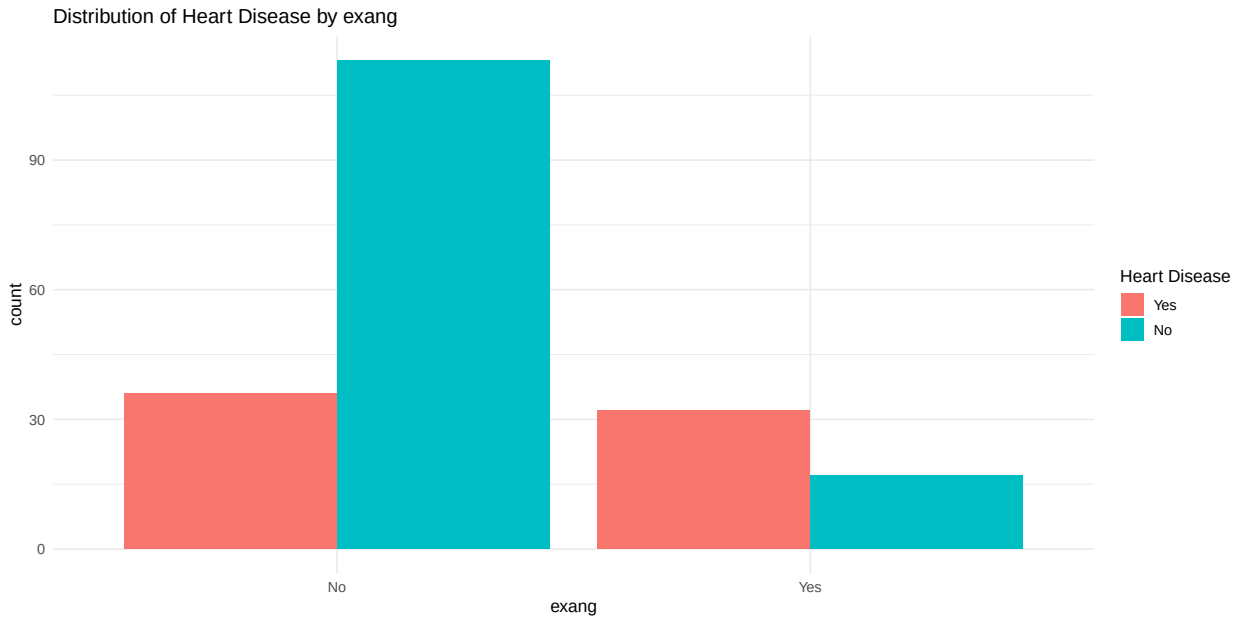
```
HeartDiseaseBar("restecg")
```



The barplot for resting ECG categories shows relatively similar distributions of heart disease across normal, wave-abnormality, and probable left ventricular hypertrophy groups. Although there are minor variations, these differences are not pronounced enough to indicate strong discriminative power. This suggests that resting ECG outcomes may provide some contextual clinical information but do not significantly differentiate between patients with and without heart disease in this dataset.

Visualize distribution of `exang`

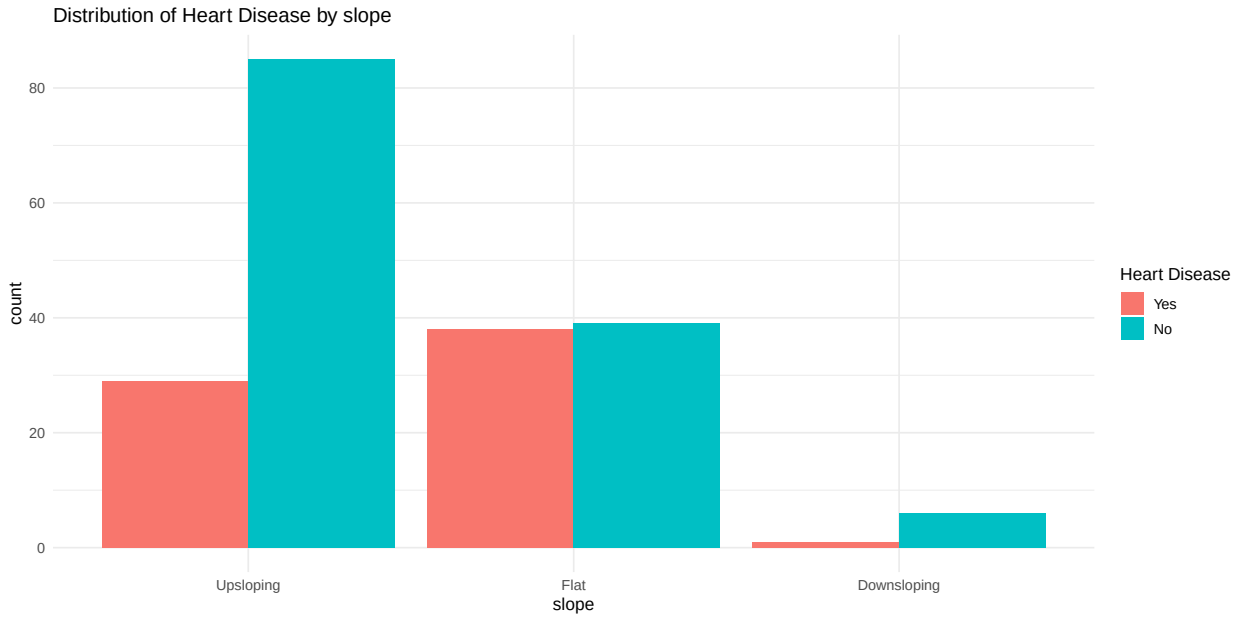
```
HeartDiseaseBar("exang")
```



Exercise-induced angina demonstrates a clear relationship with heart disease. The barplot shows that patients who experience angina during physical exertion are more likely to be diagnosed with heart disease compared to those who do not exhibit this symptom. This pattern is clinically meaningful, as exercise-induced chest pain often reflects underlying myocardial ischemia. Consequently, exang emerges as a strong and reliable predictor in identifying high-risk individuals.

Visualize distribution of `slope`

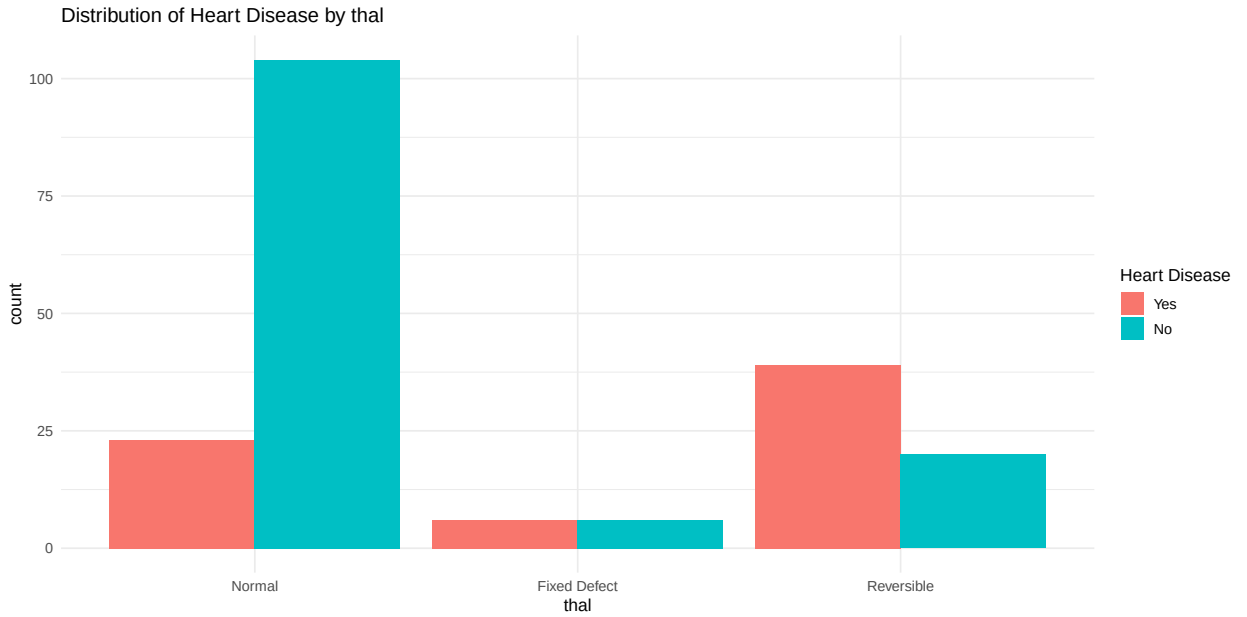
```
HeartDiseaseBar("slope")
```



The slope variable also shows substantial variation in heart disease prevalence across its categories. Patients with a flat slope (slope = 2) exhibit the highest concentration of heart disease cases, whereas the upsloping and downsloping categories show a predominance of non-diseased individuals. This distinct separation suggests that the ST segment response during exercise is a valuable clinical indicator and an important feature for predictive modeling.

Visualize distribution of `thal`

```
HeartDiseaseBar("thal")
```

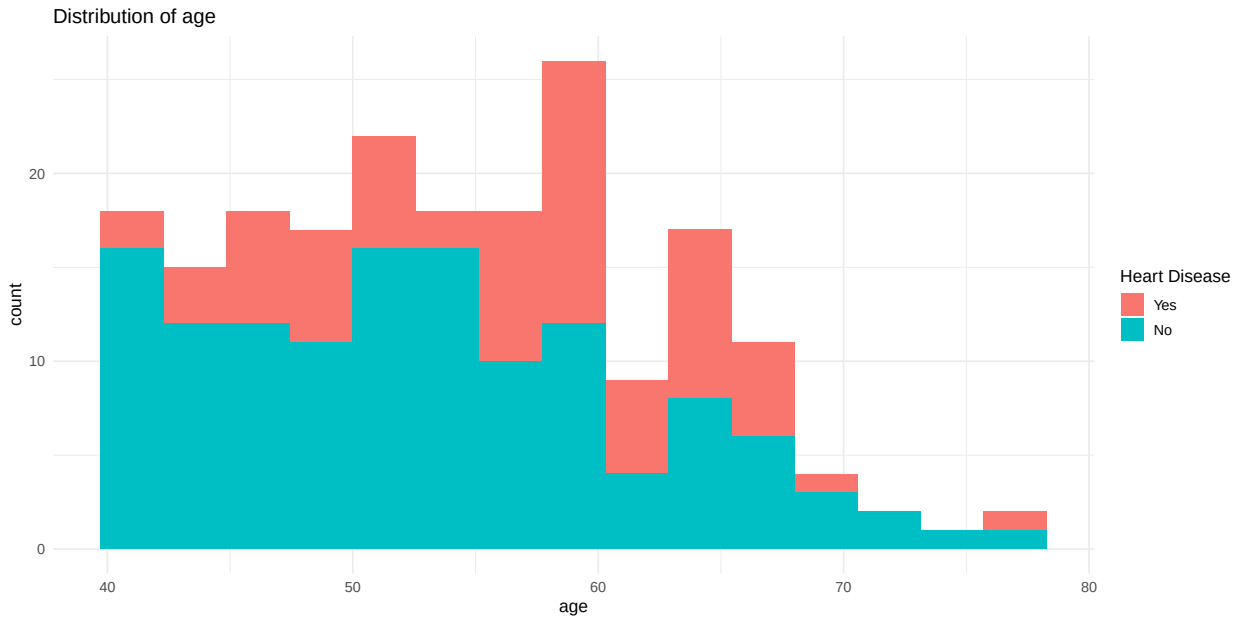


The thal variable is among the strongest categorical predictors of heart disease in the dataset. The barplot highlights that individuals with a reversible defect (thal = 7) have the highest count of heart disease cases, followed by those with fixed defects. Meanwhile, patients with normal thalassemia patterns show mostly non-diseased outcomes. This sharp contrast underscores the clinical relevance of thalassemia patterns and reinforces this variable's importance in risk prediction models.

Histograms for Numerical Variables

Histogram of `age`

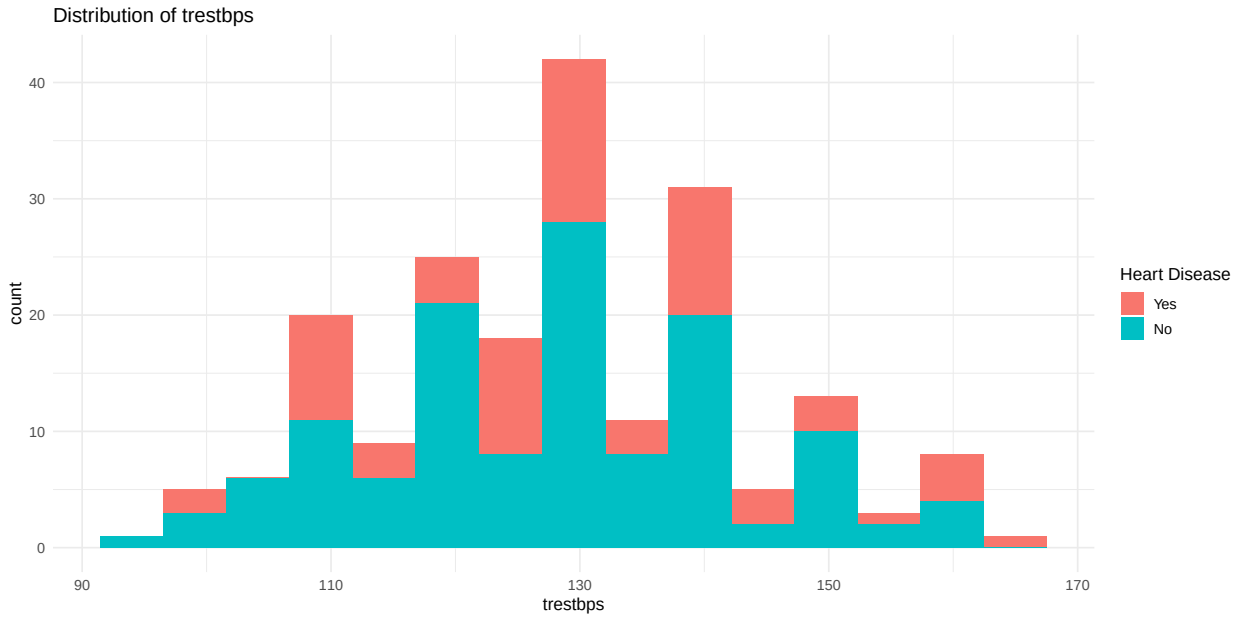
```
HeartDiseaseHist("age")
```



The age distribution shows a strong concentration of patients between their mid-40s and mid-60s. Both heart-disease and non-heart-disease groups occupy this same range, but the heart-disease bars appear more prominent in the upper-50s to upper-60s region. This suggests that while the dataset is primarily middle-aged and older adults, heart disease becomes more common as age increases. The distribution is generally uniform—with no extreme skew—indicating a broad age spread within the filtered 40+ population.

Histogram of `trestbps`

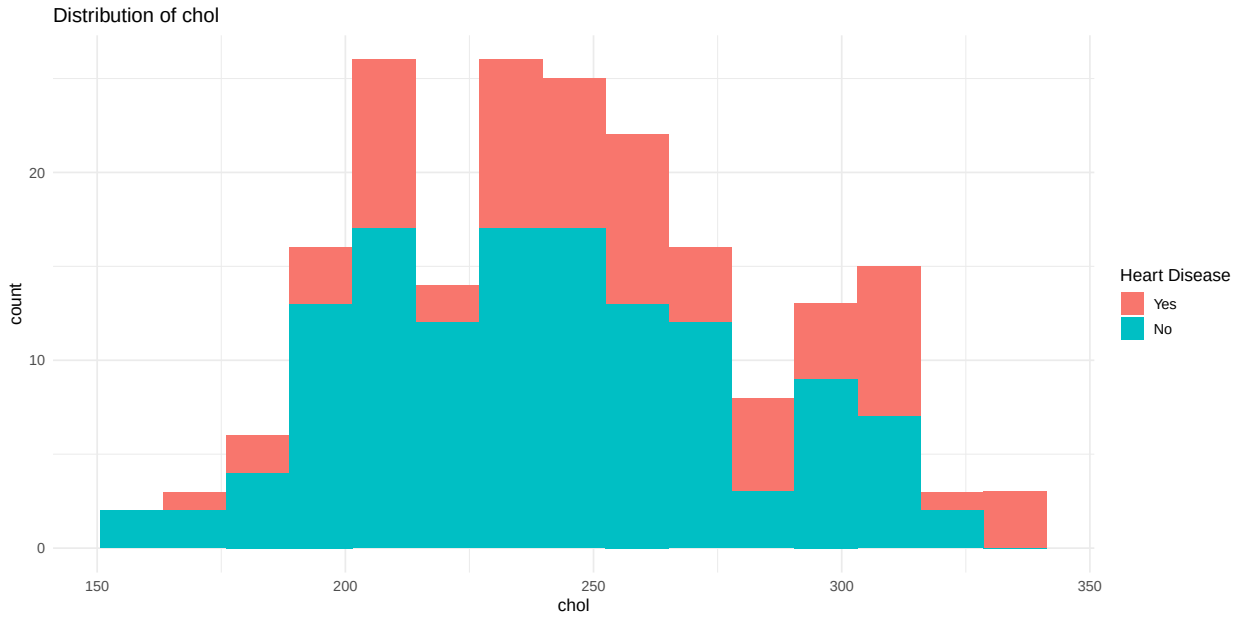
```
HeartDiseaseHist("trestbps")
```



Resting blood pressure shows a moderate spread centered around 120–140 mmHg. Heart-disease cases appear evenly mixed throughout this range, with no noticeable clustering at either extreme. The distribution does not show separation between disease groups, reinforcing that resting blood pressure—after outlier filtering—does not strongly differentiate the two populations. Slight peaks around 130–140 mmHg are visible, reflecting typical clinical norms.

Histogram of `chol`

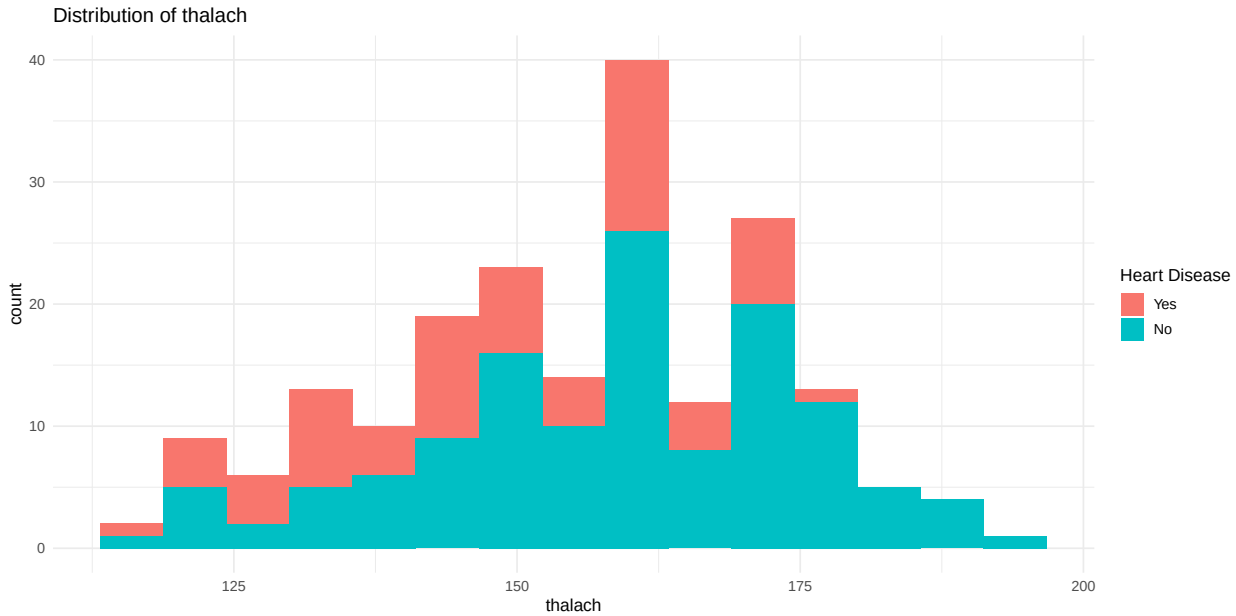
```
HeartDiseaseHist("chol")
```



Cholesterol levels range mostly between 180–300 mg/dl, with well-populated bins near 220–260 mg/dl. Both groups show similar distribution shapes, but heart-disease cases appear slightly more frequent in bins above ~240 mg/dl. This supports cholesterol as a mild-to-moderate risk indicator: elevated values are somewhat more common among those with heart disease, but the distributions still heavily overlap.

Histogram of `thalach`

```
HeartDiseaseHist("thalach")
```

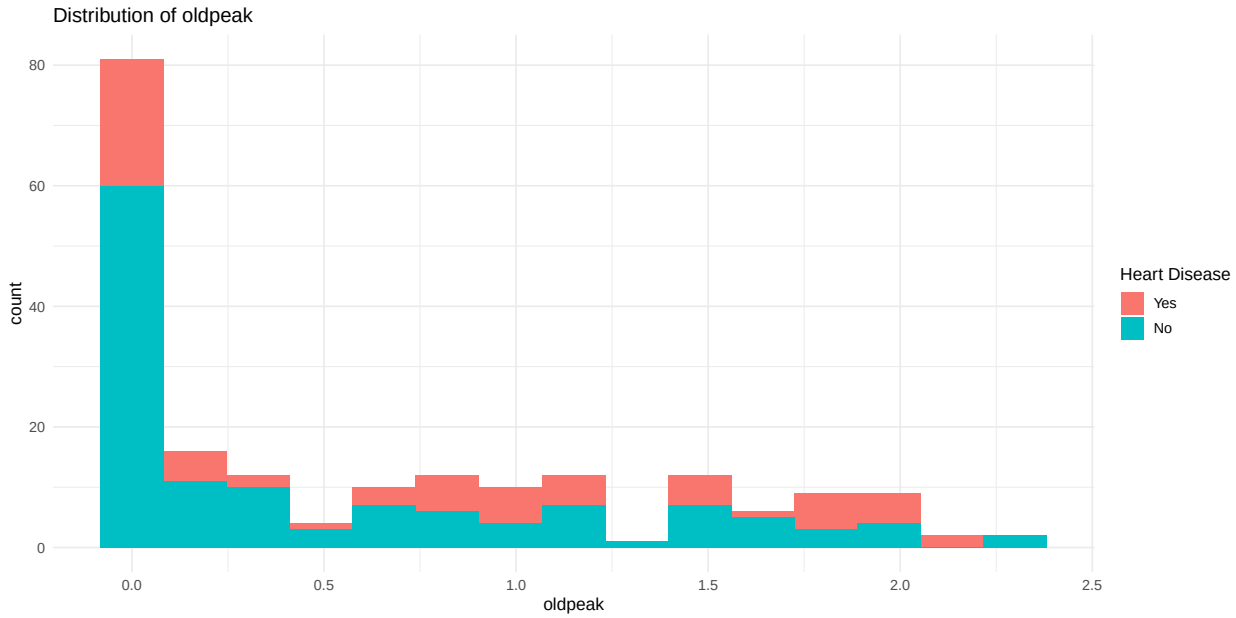


This histogram displays a noticeable distinction: patients without heart disease tend to cluster in the higher heart-rate area (150–180 bpm), while those with heart disease appear more often in the lower range

(120–150 bpm). The distribution visually slopes downward as heart-rate increases for the heart-disease group. This aligns with the clinical observation that impaired cardiovascular function can limit achievable heart rate during exercise.

Histogram of `oldpeak`

```
HeartDiseaseHist("oldpeak")
```

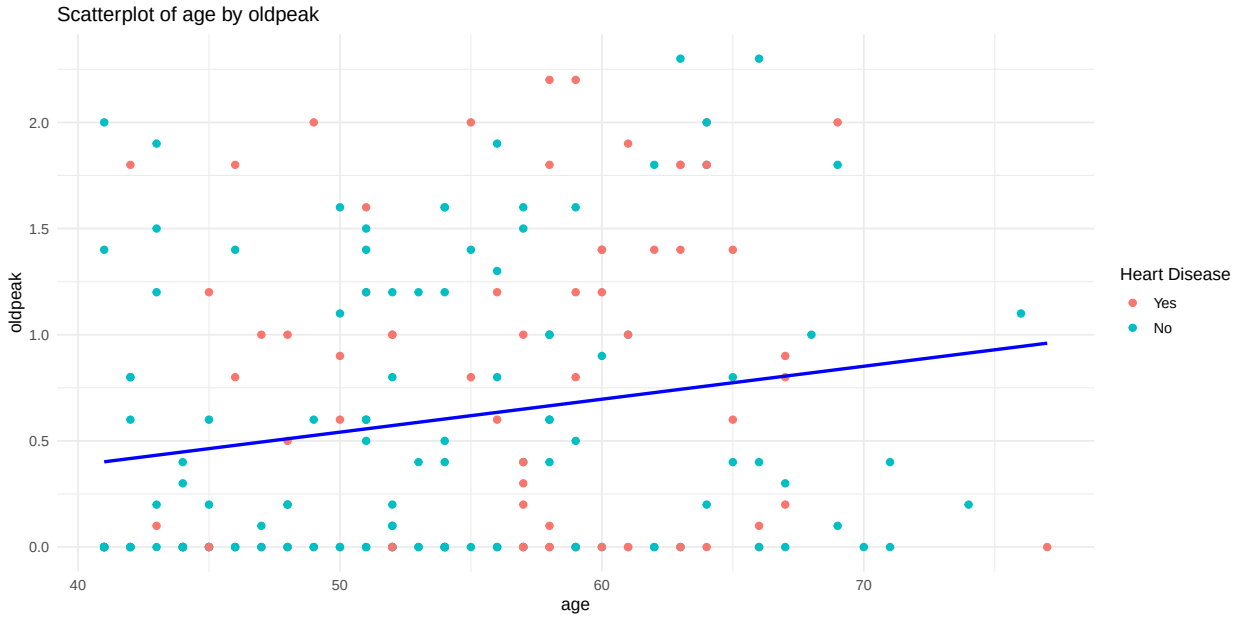


The distribution of oldpeak is heavily right-skewed, with the largest concentration near 0.0—particularly for those without heart disease. As oldpeak values increase (1.0–2.3), the proportion of heart-disease cases becomes noticeably higher. This histogram demonstrates one of the clearest separations among the numerical variables: higher ST depression during exercise is strongly associated with the presence of heart disease.

Scatterplots for numerical variables

Scatterplot of `age` vs `oldpeak`

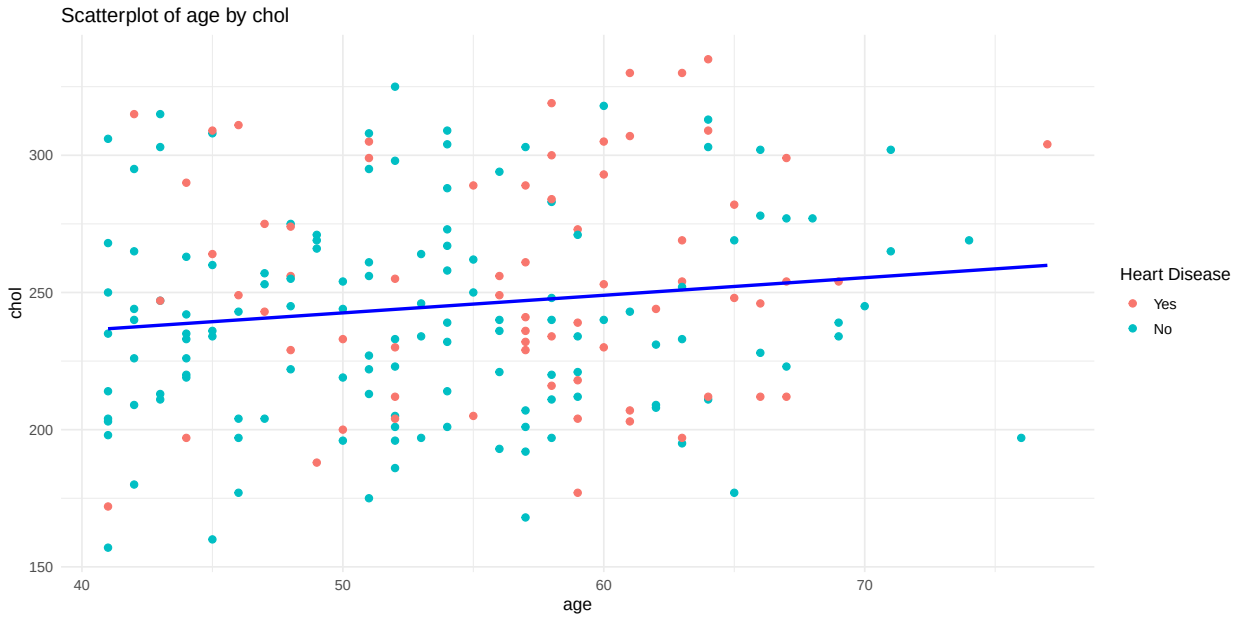
```
HeartDiseaseScatter("age", "oldpeak")
```



This scatterplot shows a mild upward trend: as age increases, oldpeak (ST depression during exercise) also tends to increase. Patients with heart disease (red) appear more concentrated in the higher oldpeak range across most age groups. Although both groups are distributed throughout the plot, the general pattern suggests that older individuals experience more ST depression—an indicator commonly associated with ischemic changes.

Scatterplot of `age` vs `chol`

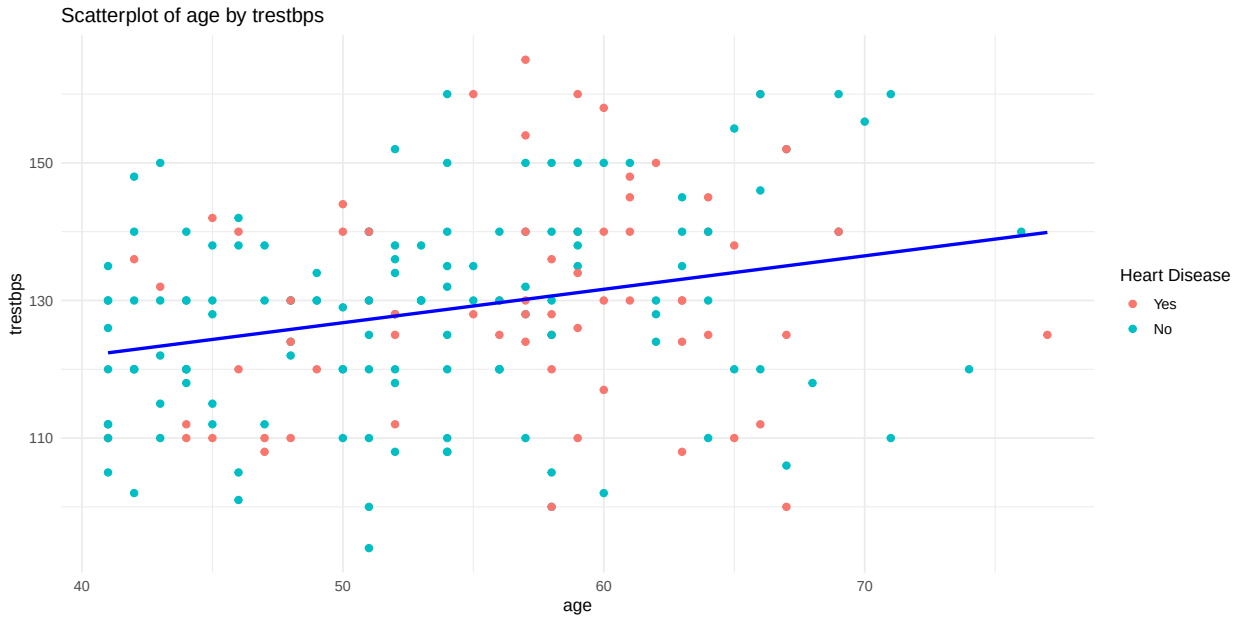
```
HeartDiseaseScatter("age", "chol")
```



Cholesterol levels appear widely scattered across ages 40–70 with no strong visual correlation. The regression line shows only a slight upward slope, implying a very weak relationship between age and cholesterol. Both "Yes" and "No" groups overlap heavily, indicating that cholesterol does not systematically rise with age in this filtered dataset. Heart-disease points do appear slightly more frequent at the higher cholesterol values, but the separation remains minimal.

Scatterplot of age vs trestbps

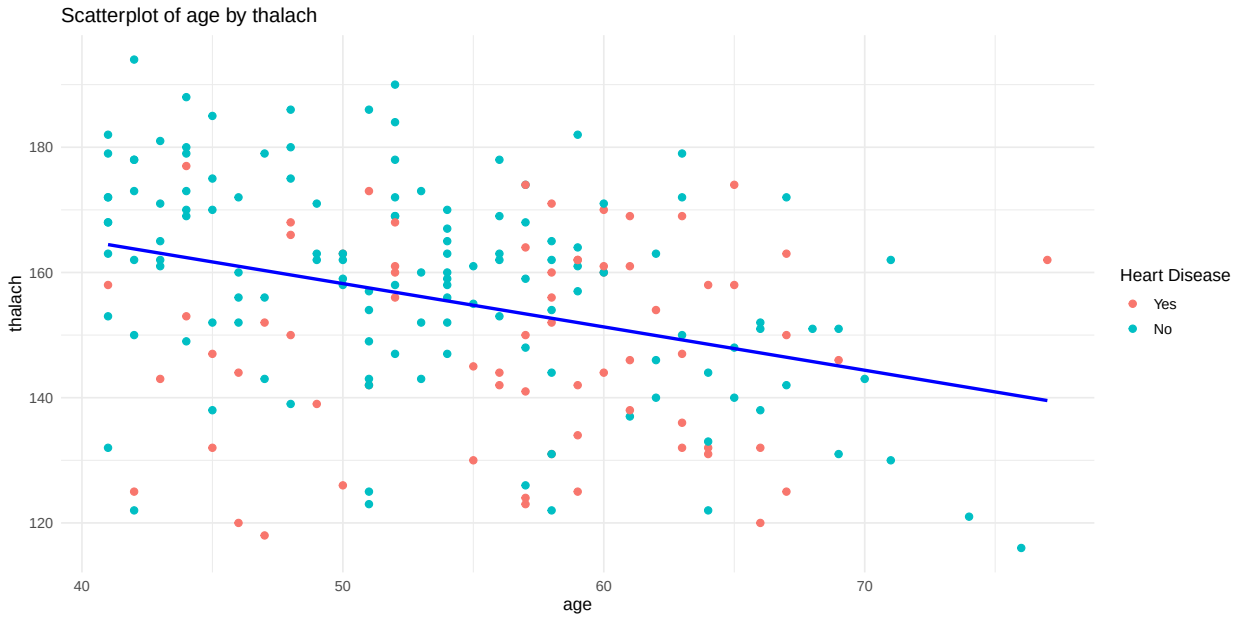
```
HeartDiseaseScatter("age", "trestbps")
```



The scatterplot shows a mild positive trend: older patients tend to have slightly higher resting blood pressure. However, similar to age–cholesterol, both groups overlap almost completely. There is no clear visual separation between heart disease and non-heart-disease patients. This supports that trestbps, after outlier removal, offers limited discriminatory power.

Scatterplot of age vs thalach

```
HeartDiseaseScatter("age", "thalach")
```

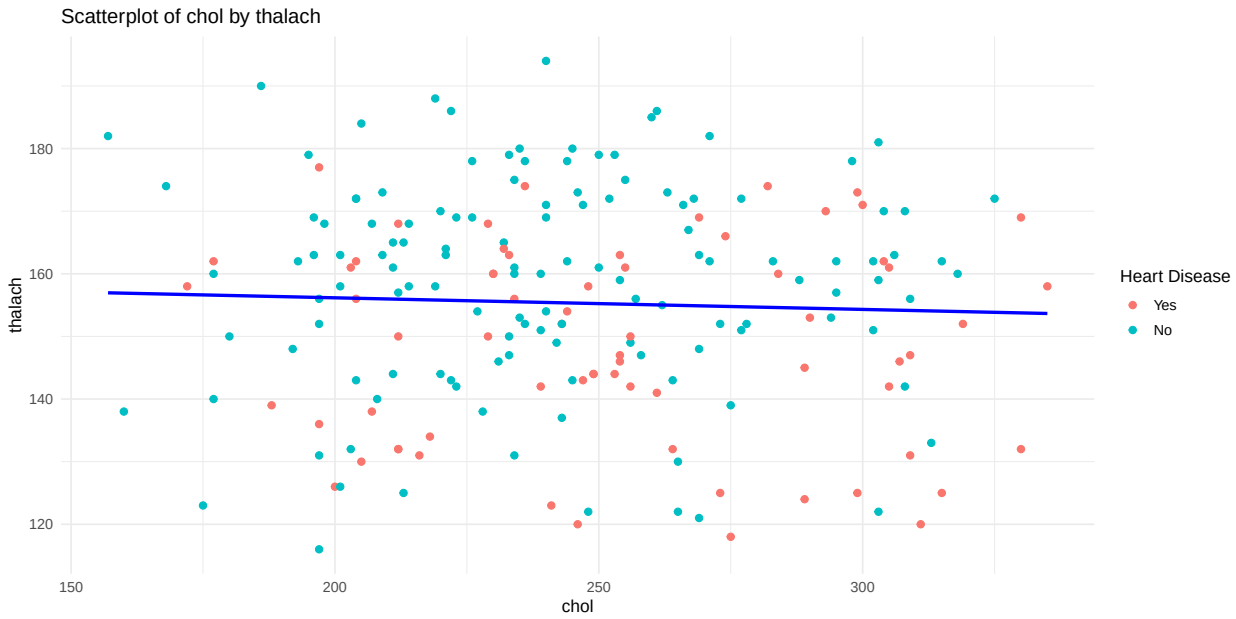


This relationship shows the clearest downward trend: as age increases, maximum heart rate achieved

decreases. The visual slope is strong and clinically expected. Importantly, heart-disease patients (red) stay more concentrated in the lower-thalach bands, while non-disease patients (blue) extend more frequently into higher heart-rate values. This scatterplot visually reinforces thalach as a valuable predictor.

Scatterplot of `chol` vs `thalach`

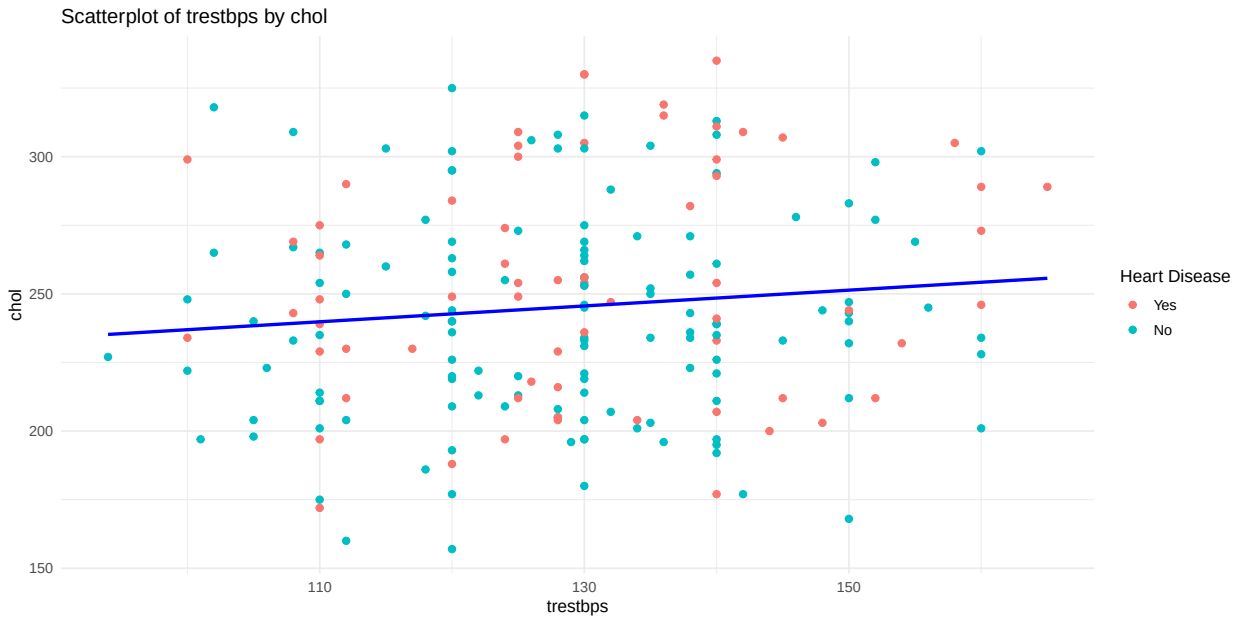
```
HeartDiseaseScatter("chol", "thalach")
```



There is a subtle negative trend: higher cholesterol is associated with slightly lower thalach values. Although both groups overlap, the "No" group extends further into the high-heart-rate range, while the "Yes" group clusters slightly lower. This indicates a soft interaction pattern: individuals with higher cholesterol may also have reduced heart-rate performance.

Scatterplot of `trestbps` vs `chol`

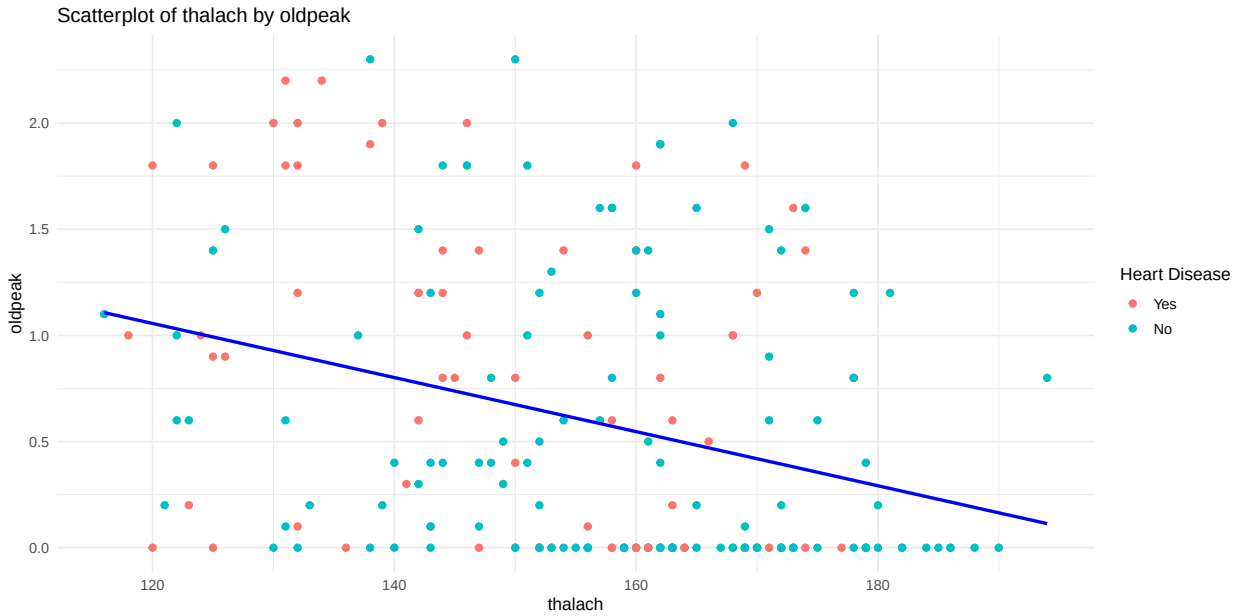
```
HeartDiseaseScatter("trestbps", "chol")
```



This scatterplot shows broad dispersion and only a faint upward slope—higher blood pressure loosely corresponds to higher cholesterol. Both outcome groups are fully intermixed with no discernible clustering. The lack of separation confirms that the two variables are not jointly informative for predicting heart disease.

Scatterplot of `thalach` vs `oldpeak`

```
HeartDiseaseScatter("thalach", "oldpeak")
```



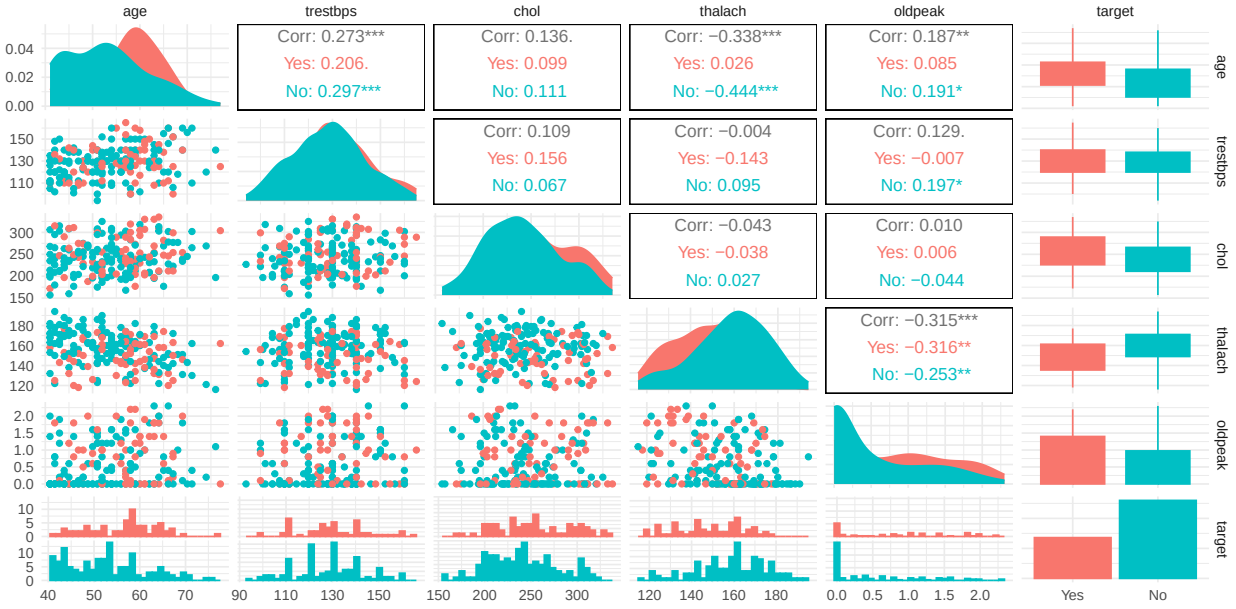
This is one of the most informative scatterplots. A clear negative relationship appears: higher maximum heart rate corresponds to lower oldpeak values. Heart-disease patients cluster more densely in the region defined by lower thalach and higher oldpeak, while non-disease patients cluster toward higher thalach and lower oldpeak. This visual pattern strongly aligns with clinical expectations: reduced heart-rate capacity and elevated ST depression during exercise are key indicators of compromised heart function.

Pairwise correlation plots

Pairwise correlation plot for numerical variables

```
ggpairs(Heart.df[, c("age", "trestbps", "chol", "thalach", "oldpeak", "target")],
        aes(color = target, fill = target))
```

```
## `stat_bin()` using `bins = 30`. Pick better value `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value `binwidth`.
```



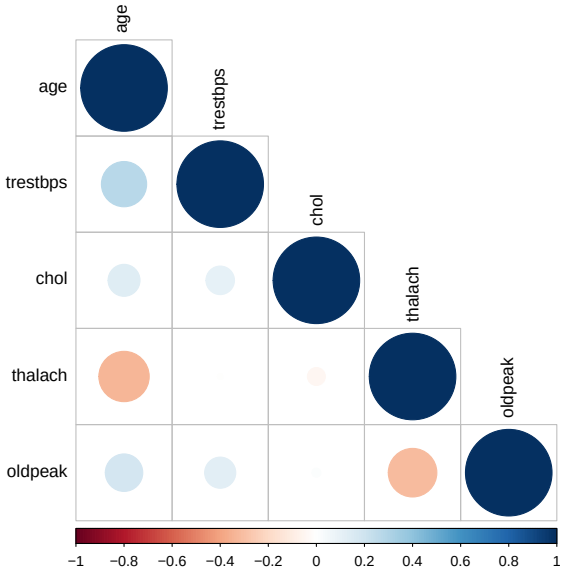
Correlation matrix

Correlation matrix for numerical variables

```
# Selecting only continuous variables
continuous_vars <- c("age", "trestbps", "chol", "thalach", "oldpeak")
continuous_data <- Heart.df %>% select(all_of(continuous_vars))

# Calculating correlation matrix
correlation_matrix <- cor(continuous_data)

# Plotting the correlation matrix
corrplot(correlation_matrix, method = "circle",
         type = "lower", tl.col = "black")
```


age
trestbps
chol
thalach
oldpeak
−1
−0.8
−0.6
−0.4
−0.2
0
0.2
0.4
0.6
0.8
1

Modeling

Spliting the dataset

```
# Ensure target is a factor with "Yes" as the positive class
Heart.df$target <- factor(Heart.df$target, levels = c("Yes", "No"))

# 70/30 train-test split, stratified by target
indx <- createDataPartition(Heart.df$target, p = 0.7, list = FALSE)
train_data <- Heart.df[indx, ]
test_data  <- Heart.df[-indx, ]

# Scale numeric columns
numeric_columns <- c("age", "trestbps", "chol", "thalach", "oldpeak")
train_data[ , numeric_columns] <- scale(train_data[ , numeric_columns])
test_data[  , numeric_columns] <- scale(test_data[  , numeric_columns])

# Make sure factor levels are consistent across splits
train_data$target <- factor(train_data$target, levels = c("Yes", "No"))
test_data$target  <- factor(test_data$target,  levels = c("Yes", "No"))
```

Cross-Validation Setup

```
set.seed(123)

ctrl <- trainControl(
  method          = "repeatedcv",
  number          = 5,          # 5-fold CV
  repeats         = 3,          # repeated 3 times
  classProbs      = TRUE,       # needed for ROC
  summaryFunction = twoClassSummary,
  savePredictions = "final"
)
```

Logistic Regression (Baseline Model)

```
set.seed(123)

logit_model <- train(
  target ~ .,
  data      = train_data,
  method    = "glm",
  family    = binomial,
  trControl = ctrl,
  metric    = "ROC"
)
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
logit_model
```

```
## Generalized Linear Model
##
## 139 samples
##  13 predictor
##   2 classes: 'Yes', 'No'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold, repeated 3 times)
## Summary of sample sizes: 111, 112, 111, 111, 111, 110, ...
## Resampling results:
##
##   ROC        Sens       Spec
##   0.7993784  0.6103704  0.7982456
```

Random Forest (Nonlinear Ensemble)

```
set.seed(123)

rf_model <- train(
  target ~ .,
  data      = train_data,
  method    = "rf",
  trControl = ctrl,
  metric    = "ROC",
  tuneLength = 10
)

rf_model
```

```
## Random Forest
##
## 139 samples
##  13 predictor
##   2 classes: 'Yes', 'No'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold, repeated 3 times)
## Summary of sample sizes: 111, 112, 111, 111, 111, 110, ...
## Resampling results across tuning parameters:
##
##   mtry  ROC        Sens       Spec
##    2    0.8610959  0.5814815  0.9189084
##    3    0.8581330  0.5881481  0.9118908
##    5    0.8497228  0.5888889  0.8976608
##    7    0.8416569  0.5881481  0.9009747
##    9    0.8413624  0.6103704  0.8900585
##   10    0.8401830  0.6037037  0.8752437
##   12    0.8359909  0.5837037  0.8791423
##   14    0.8335034  0.5911111  0.8717349
```

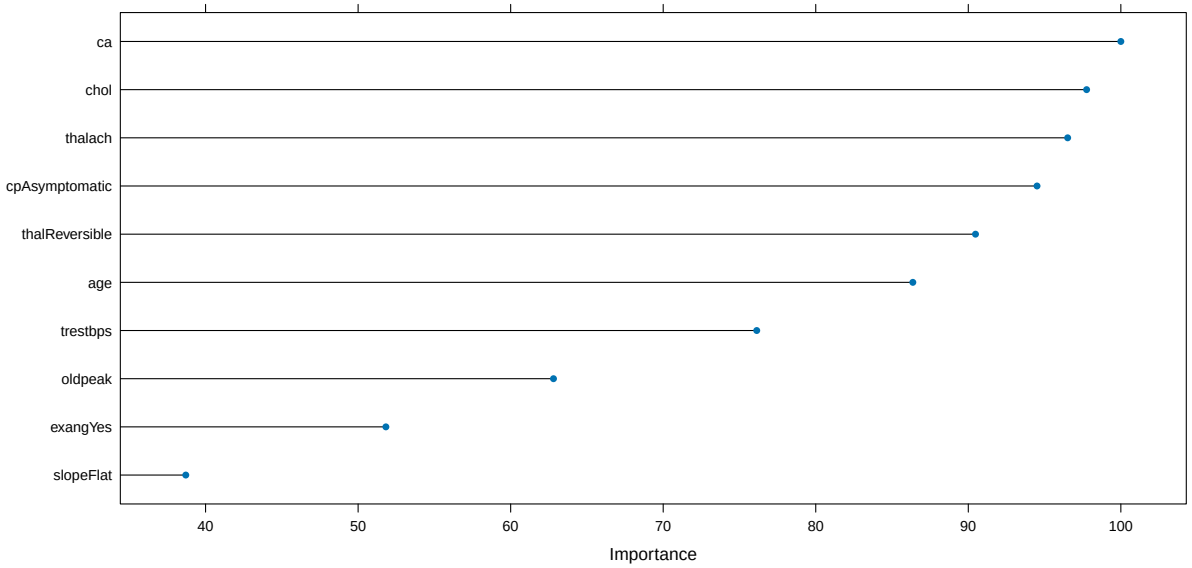
```
##   16    0.8352426  0.5977778  0.8791423
##   18    0.8317013  0.6251852  0.8643275
##
## ROC was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 2.
```

Variable importance for interpretation

```
set.seed(123)
rf_varimp <- varImp(rf_model, scale = TRUE)
rf_varimp
```

```
## rf variable importance
##
##                          Overall
## ca                       100.000
## chol                      97.756
## thalach                   96.514
## cpAsymptomatic            94.506
## thalReversible            90.473
## age                       86.364
## trestbps                  76.132
## oldpeak                   62.808
## exangYes                  51.820
## slopeFlat                 38.703
## cpNon-Angina              37.096
## sexMale                   36.219
## restecgProbable           29.738
## cpAtypical Angina         18.545
## thalFixed Defect          12.228
## fbsTrue                    8.356
## slopeDownsloping           2.540
## restecgWave-abnormality    0.000
```

```
plot(rf_varimp, top = 10)
```



XGBoost (Gradient Boosted Trees)

```
set.seed(123)

# A modest tuning grid for XGBoost
xgb_grid <- expand.grid(
  nrounds          = c(50, 100, 150),
  max_depth        = c(2, 4, 6),
  eta              = c(0.05, 0.1, 0.3),
  gamma            = 0,
  colsample_bytree = 0.8,
  min_child_weight = 1,
  subsample        = 0.8
)

set.seed(123)
xgb_model <- suppressWarnings(
  train(
    target ~ .,
    data      = train_data,
    method    = "xgbTree",
    trControl = ctrl,
    metric    = "ROC",
    tuneGrid  = xgb_grid
  )
)
```

```
## [23:12:01] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:12:01] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:12:01] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:12:01] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
```

```
## [23:12:15] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:12:15] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:12:15] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:12:15] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:13:12] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:13:12] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:13:12] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:13:12] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:13:22] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:13:22] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:13:22] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:13:22] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:14:20] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:14:20] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:14:21] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:14:21] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:14:40] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:14:40] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:14:40] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:14:40] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:14:43] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:14:43] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:14:43] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:14:43] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:14:54] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:14:54] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:14:54] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:14:54] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:15:00] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:15:00] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:15:00] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:15:00] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:15:03] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:15:03] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:15:03] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:15:03] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:15:08] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:15:08] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:15:08] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:15:08] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:15:12] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:15:12] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:15:12] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:15:12] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:15:18] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:15:18] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:15:19] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:15:19] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:15:43] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:15:43] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:15:43] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:15:43] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:15:48] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:15:48] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
```

```
## [23:15:48] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:15:48] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:15:51] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:15:51] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:15:51] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:15:51] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:15:53] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:15:53] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:15:53] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:15:53] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:15:55] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:15:55] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:15:55] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:15:55] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:15:57] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:15:57] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:15:57] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:15:57] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:15:58] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:15:58] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:15:58] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:15:58] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:16:01] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:16:01] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:16:01] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:16:01] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:16:02] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:16:02] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:16:02] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:16:02] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:16:07] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:16:07] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:16:07] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:16:08] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:16:10] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:16:10] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:16:10] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:16:10] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:16:11] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:16:11] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:16:11] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:16:11] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:16:11] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:16:11] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:16:11] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:16:12] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:16:12] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:16:12] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:16:12] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:16:12] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:16:13] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:16:13] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:16:13] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:16:13] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
```

```
## [23:16:14] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:16:14] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:16:14] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:16:14] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:16:16] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:16:16] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:16:16] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:16:16] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:16:17] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:16:17] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:16:17] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:16:17] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:16:19] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:16:19] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:16:19] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:16:19] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:16:22] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:16:22] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:16:22] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:16:22] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:16:24] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:16:24] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:16:24] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:16:24] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:16:26] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:16:26] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:16:26] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:16:26] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:16:28] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:16:28] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:16:28] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:16:28] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:16:29] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:16:29] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:16:29] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:16:29] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:16:30] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:16:30] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:16:30] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:16:31] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:16:32] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:16:32] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:16:32] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:16:32] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:16:35] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:16:35] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:16:35] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:16:35] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:16:38] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:16:38] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:16:38] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:16:38] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:16:40] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:16:40] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
```

```
## [23:16:40] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:16:40] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:16:41] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:16:41] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:16:41] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:16:41] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:16:43] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:16:43] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:16:43] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:16:43] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:16:45] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:16:45] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:16:45] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:16:45] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:16:47] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:16:47] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:16:47] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:16:47] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:16:50] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:16:50] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:16:50] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:16:50] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:16:53] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:16:53] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:16:53] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:16:53] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:16:55] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:16:55] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:16:55] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:16:55] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:16:58] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:16:58] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:16:58] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:16:58] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:01] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:01] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:01] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:01] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:02] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:02] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:02] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:02] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:04] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:04] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:04] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:04] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:06] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:06] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:06] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:06] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:07] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:07] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:08] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:08] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
```

```
## [23:17:10] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:10] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:10] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:10] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:13] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:13] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:13] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:13] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:14] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:14] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:14] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:14] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:15] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:15] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:15] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:15] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:15] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:15] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:15] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:15] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:16] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:16] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:16] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:16] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:17] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:17] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:17] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:18] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:19] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:19] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:19] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:19] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:20] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:20] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:20] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:20] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:22] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:22] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:22] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:22] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:24] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:24] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:24] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:24] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:25] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:25] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:25] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:25] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:27] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:27] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:27] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:27] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:28] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:28] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
```

```
## [23:17:28] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:28] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:30] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:30] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:30] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:30] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:34] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:34] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:34] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:34] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:37] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:37] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:37] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:37] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:40] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:40] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:40] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:40] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:42] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:42] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:42] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:42] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:45] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:45] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:45] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:45] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:46] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:46] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:46] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:46] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:48] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:48] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:48] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:48] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:51] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:51] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:51] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:51] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:52] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:52] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:52] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:52] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:52] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:52] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:52] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:52] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:53] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:53] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:53] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:53] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:54] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:54] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:54] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:54] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
```

```
## [23:17:55] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:55] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:55] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:17:55] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:18:34] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:18:34] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:18:34] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:18:34] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:18:43] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:18:43] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:18:43] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:18:43] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:18:49] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:18:49] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:18:49] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:18:49] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:18:53] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:18:53] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:18:53] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:18:53] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:18:55] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:18:55] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:18:55] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:18:55] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:18:57] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:18:57] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:18:57] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:18:57] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:19:00] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:19:00] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:19:00] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:19:00] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:19:01] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:19:01] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:19:01] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:19:01] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:19:03] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:19:03] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:19:03] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:19:03] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:19:05] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:19:05] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:19:05] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:19:05] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:19:06] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:19:06] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:19:06] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:19:07] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:19:09] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:19:09] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:19:09] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:19:09] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:19:11] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:19:11] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
```

```
## [23:19:11] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:19:11] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:19:12] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:19:12] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:19:12] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:19:12] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:19:13] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:19:13] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:19:13] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:19:13] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:19:15] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:19:15] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:19:15] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:19:15] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:19:20] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:19:20] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:19:21] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:19:21] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:19:25] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:19:25] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:19:25] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:19:25] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:19:31] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:19:31] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:19:31] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:19:31] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:19:35] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:19:35] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:19:35] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:19:35] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:19:42] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:19:42] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:19:42] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:19:42] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:19:48] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:19:48] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:19:48] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:19:48] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:19:51] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:19:51] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:19:51] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:19:51] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:19:53] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:19:53] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:19:53] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:19:53] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:19:57] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:19:57] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:19:57] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:19:57] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:20:02] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:20:02] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:20:02] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:20:02] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
```

```
## [23:20:10] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:20:10] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:20:10] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:20:10] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:20:20] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:20:20] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:20:20] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:20:20] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:20:22] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:20:23] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:20:23] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:20:23] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:20:26] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:20:26] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:20:26] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:20:26] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:20:28] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:20:28] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:20:28] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:20:28] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:20:29] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:20:29] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:20:29] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:20:29] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:20:33] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:20:33] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:20:33] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:20:33] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:21:19] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:21:19] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:21:19] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:21:19] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:21:22] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:21:22] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:21:22] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:21:22] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:21:25] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:21:25] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:21:25] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:21:25] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:21:27] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:21:27] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:21:27] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:21:27] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:21:49] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:21:49] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:21:49] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:21:49] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:21:55] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:21:55] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:21:55] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:21:55] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:21:56] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:21:56] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
```

```
## [23:21:56] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:21:56] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:21:57] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:21:57] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:21:57] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:21:57] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:21:58] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:21:58] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:21:58] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:21:58] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:22:00] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:22:01] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:22:01] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:22:01] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:22:02] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:22:02] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:22:02] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:22:02] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:22:06] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:22:06] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:22:06] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:22:06] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:22:08] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:22:08] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:22:08] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:22:08] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:22:09] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:22:09] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:22:09] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:22:09] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:22:12] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:22:12] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:22:12] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:22:12] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:22:15] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:22:15] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:22:15] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:22:15] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:22:16] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:22:16] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:22:16] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:22:16] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:22:16] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:22:16] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:22:16] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:22:16] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:22:19] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:22:19] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:22:20] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [23:22:20] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
```

```
xgb_model
```

```
## eXtreme Gradient Boosting
```

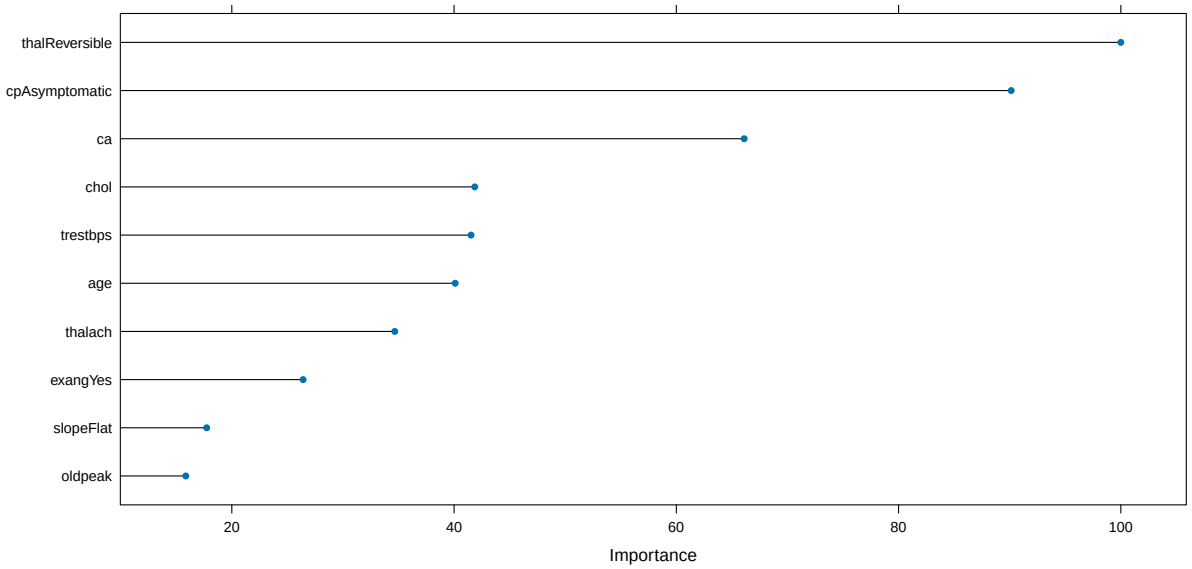
```
##
## 139 samples
##  13 predictor
##   2 classes: 'Yes', 'No'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold, repeated 3 times)
## Summary of sample sizes: 111, 112, 111, 111, 111, 110, ...
## Resampling results across tuning parameters:
##
##   eta   max_depth  nrounds  ROC        Sens       Spec
##   0.05  2           50      0.8597163  0.6229630  0.9153996
##   0.05  2          100      0.8515313  0.6088889  0.8826511
##   0.05  2          150      0.8481980  0.6222222  0.8898635
##   0.05  4           50      0.8616872  0.6096296  0.9044834
##   0.05  4          100      0.8567750  0.5962963  0.8898635
##   0.05  4          150      0.8515660  0.6037037  0.8935673
##   0.05  6           50      0.8623868  0.6170370  0.8933723
##   0.05  6          100      0.8580204  0.6029630  0.8970760
##   0.05  6          150      0.8526121  0.6096296  0.8972710
##   0.10  2           50      0.8494715  0.6229630  0.8896686
##   0.10  2          100      0.8399199  0.6081481  0.8752437
##   0.10  2          150      0.8335824  0.6074074  0.8606238
##   0.10  4           50      0.8553541  0.6318519  0.8937622
##   0.10  4          100      0.8484319  0.6511111  0.8608187
##   0.10  4          150      0.8420338  0.6303704  0.8645224
##   0.10  6           50      0.8597682  0.6103704  0.8935673
##   0.10  6          100      0.8498159  0.6029630  0.8935673
##   0.10  6          150      0.8402556  0.6237037  0.8865497
##   0.30  2           50      0.8214858  0.6081481  0.8458090
##   0.30  2          100      0.8145896  0.6140741  0.8458090
##   0.30  2          150      0.8157592  0.6148148  0.8307992
##   0.30  4           50      0.8369482  0.6303704  0.8721248
##   0.30  4          100      0.8345008  0.6437037  0.8719298
##   0.30  4          150      0.8340069  0.6503704  0.8610136
##   0.30  6           50      0.8428243  0.6237037  0.8680312
##   0.30  6          100      0.8439744  0.6237037  0.8571150
##   0.30  6          150      0.8429456  0.6311111  0.8606238
##
## Tuning parameter 'gamma' was held constant at a value of 0
## Tuning
##
## Tuning parameter 'min_child_weight' was held constant at a value of 1
##
## Tuning parameter 'subsample' was held constant at a value of 0.8
## ROC was used to select the optimal model using the largest value.
## The final values used for the model were nrounds = 50, max_depth = 6, eta
##  = 0.05, gamma = 0, colsample_bytree = 0.8, min_child_weight = 1 and
##  subsample = 0.8.
```

Variable importance

```
set.seed(123)
xgb_varimp <- varImp(xgb_model, scale = TRUE)
xgb_varimp
```

```
## xgbTree variable importance
##
##                           Overall
## thalReversible            100.000
## cpAsymptomatic             90.132
## ca                         66.102
## chol                       41.863
## trestbps                   41.527
## age                        40.100
## thalach                    34.665
## exangYes                   26.415
## slopeFlat                  17.740
## oldpeak                    15.857
## sexMale                    13.137
## restecgProbable            12.975
## cpNon-Angina                6.243
## thalFixed Defect            5.888
## cpAtypical Angina           2.684
## fbsTrue                     0.000
## slopeDownsloping            0.000
## restecgWave-abnormality     0.000
```

```
plot(xgb_varimp, top = 10)
```



Compare Models via Resampling (CV Results)

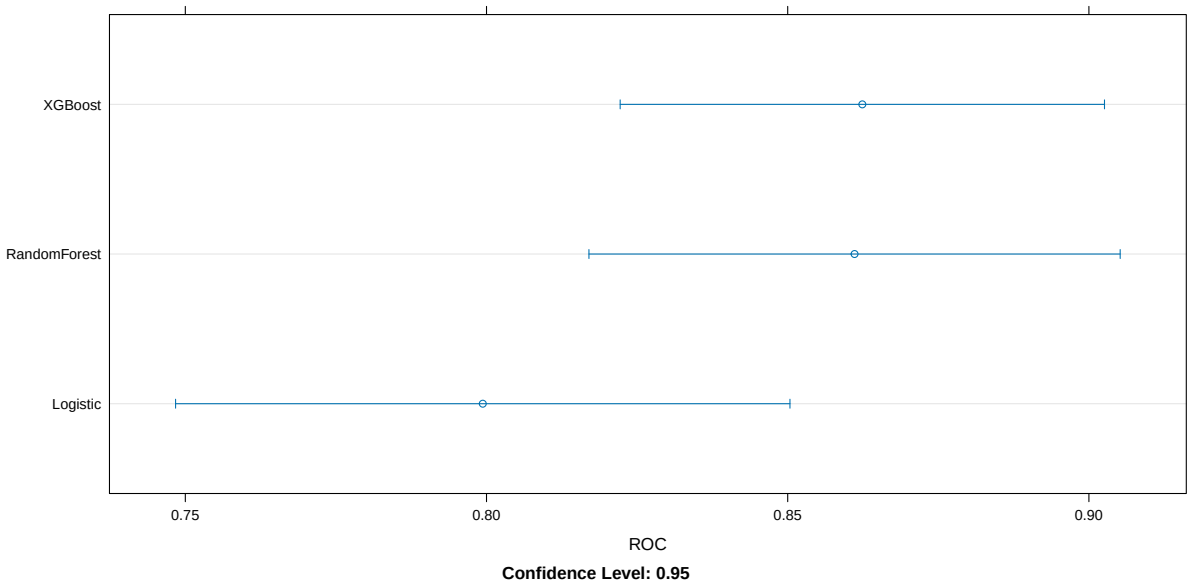
```
model_list <- list(
  Logistic      = logit_model,
  RandomForest  = rf_model,
  XGBoost       = xgb_model
)

resamps <- resamples(model_list)
summary(resamps)
```

```
##
## Call:
## summary.resamples(object = resamps)
##
## Models: Logistic, RandomForest, XGBoost
## Number of resamples: 15
##
## ROC
##                   Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## Logistic     0.5368421 0.7592593 0.8105263 0.7993784 0.8666667 0.9012346    0
## RandomForest 0.7160494 0.8115741 0.8859649 0.8610959 0.9227339 0.9691358    0
## XGBoost      0.7469136 0.8188434 0.8666667 0.8623868 0.9029240 0.9722222    0
##
## Sens
##              Min.   1st Qu. Median      Mean   3rd Qu. Max. NA's
## Logistic      0.2 0.5555556    0.6 0.6103704 0.7000000  0.9    0
## RandomForest  0.3 0.4722222    0.6 0.5814815 0.6833333  0.9    0
## XGBoost       0.3 0.5277778    0.6 0.6170370 0.7000000  0.9    0
##
## Spec
##                   Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## Logistic     0.6111111 0.7777778 0.7777778 0.7982456 0.8377193 0.9444444    0
## RandomForest 0.7222222 0.8888889 0.9444444 0.9189084 0.9473684 1.0000000    0
## XGBoost      0.7222222 0.8654971 0.8888889 0.8933723 0.9444444 1.0000000    0
```

Visual comparison of ROC across models

```
dotplot(resamps, metric = "ROC")
```


XGBoost
RandomForest
Logistic
0.75
0.80
0.85
0.90
ROC
Confidence Level: 0.95

Evaluation

Helper Function

Helper function to compute confusion matrix & AUC

```
# Helper function to compute confusion matrix & AUC
eval_model <- function(model, test_data, positive = "Yes") {
  # Class predictions
  pred_class <- predict(model, newdata = test_data)

  # Probabilities for positive class
  pred_prob <- predict(model, newdata = test_data, type = "prob")[ , positive]

  # Confusion matrix
  cm <- confusionMatrix(pred_class, test_data$target, positive = positive)

  # ROC and AUC
  roc_obj <- roc(response = test_data$target,
                 predictor = pred_prob,
                 levels   = c("No", "Yes"))
  auc_val <- auc(roc_obj)

  list(
    confusion = cm, auc = auc_val
  )
}
```

Evaluate each model

Evaluate each model on the test set

```
logit_eval <- eval_model(logit_model, test_data)
```

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

```
rf_eval    <- eval_model(rf_model, test_data)
```

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

```
xgb_eval <- suppressWarnings(eval_model(xgb_model, test_data))
```

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

```
# Inspect Results
logit_eval$confusion
```

```
## Confusion Matrix and Statistics
##
```

```
##           Reference
## Prediction Yes No
##        Yes  13  7
##        No    7 32
##
##                Accuracy : 0.7627
##                  95% CI : (0.6341, 0.8638)
##     No Information Rate : 0.661
##     P-Value [Acc > NIR] : 0.0622
##
##                   Kappa : 0.4705
##
##  Mcnemar's Test P-Value : 1.0000
##
##             Sensitivity : 0.6500
##             Specificity : 0.8205
##          Pos Pred Value : 0.6500
##          Neg Pred Value : 0.8205
##              Prevalence : 0.3390
##          Detection Rate : 0.2203
##    Detection Prevalence : 0.3390
##       Balanced Accuracy : 0.7353
##
##        'Positive' Class : Yes
##
```

```
logit_eval$auc
```

```
## Area under the curve: 0.7974
```

```
rf_eval$confusion
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Yes No
##        Yes  11  2
##        No    9 37
##
##                Accuracy : 0.8136
##                  95% CI : (0.6909, 0.9031)
##     No Information Rate : 0.661
##     P-Value [Acc > NIR] : 0.007536
##
##                   Kappa : 0.5452
##
##  Mcnemar's Test P-Value : 0.070440
##
##             Sensitivity : 0.5500
##             Specificity : 0.9487
##          Pos Pred Value : 0.8462
##          Neg Pred Value : 0.8043
##              Prevalence : 0.3390
```

```
##          Detection Rate : 0.1864
##    Detection Prevalence : 0.2203
##       Balanced Accuracy : 0.7494
##
##        'Positive' Class : Yes
##
```

```
rf_eval$auc
```

```
## Area under the curve: 0.8756
```

```
xgb_eval$confusion
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Yes No
##        Yes  10  5
##        No   10 34
##
##                Accuracy : 0.7458
##                  95% CI : (0.6156, 0.8502)
##     No Information Rate : 0.661
##     P-Value [Acc > NIR] : 0.1062
##
##                   Kappa : 0.3959
##
##  Mcnemar's Test P-Value : 0.3017
##
##             Sensitivity : 0.5000
##             Specificity : 0.8718
##          Pos Pred Value : 0.6667
##          Neg Pred Value : 0.7727
##              Prevalence : 0.3390
##          Detection Rate : 0.1695
##    Detection Prevalence : 0.2542
##       Balanced Accuracy : 0.6859
##
##        'Positive' Class : Yes
##
```

```
xgb_eval$auc
```

```
## Area under the curve: 0.8526
```

Compact Test-Set Performance Summary

```
test_summary <- data.frame(
  Model = c("Logistic Regression", "Random Forest", "XGBoost"),
```

```
  AUC   = c(
    as.numeric(logit_eval$auc),
    as.numeric(rf_eval$auc),
    as.numeric(xgb_eval$auc)
  ),
  Accuracy = c(
    logit_eval$confusion$overall["Accuracy"],
    rf_eval$confusion$overall["Accuracy"],
    xgb_eval$confusion$overall["Accuracy"]
  )
)

test_summary
```

```
##                 Model       AUC  Accuracy
## 1 Logistic Regression 0.7974359 0.7627119
## 2       Random Forest 0.8756410 0.8135593
## 3             XGBoost 0.8525641 0.7457627
```