Predicting Cardiovascular disease and its risk factor using machine learning algorithms

Varun Putta, Zhiyi Ying, Hang Lei

2024-04-27

```
#Install and load all important libraries
suppressMessages(install.packages("tidymodels", repos = "http://cran-us.project.com", quietly = TRUE))
```

```
## Warning: unable to access index for repository http://cran-us.project.com/src/contrib:
##   cannot open URL 'http://cran-us.project.com/src/contrib/PACKAGES'
```

```
## Warning: package 'tidymodels' is not available for this version of R
##
## A version of this package for your version of R might be available elsewhere,
## see the ideas at
## https://cran.r-project.org/doc/manuals/r-patched/R-admin.html#Installing-packages
```

```
## Warning: unable to access index for repository http://cran-us.project.com/bin/macosx/big-sur-arm64/c
##   cannot open URL 'http://cran-us.project.com/bin/macosx/big-sur-arm64/contrib/4.3/PACKAGES'
```

```
suppressMessages(library(dplyr))
suppressMessages(library(ggplot2))
suppressMessages(library(tidymodels))
suppressMessages(library(readr))
suppressMessages(library(ISLR))
suppressMessages(library(tidyverse))
suppressMessages(library(caret))
suppressMessages(library(rpart))
suppressMessages(library(rpart.plot))
suppressMessages(library(ROCR))
```

Our Research Question is : How can a novel machine learning model, integrating socio-economic indicators, environmental factors, and traditional risk factors, enhance the accuracy of predicting cardiovascular disease risk compared to existing models primarily reliant on traditional risk factors?

Outcome Varibale : Heart Disease

Data Preparation

One essential part of this project is to import and clean the data as needed. According to the World Health Organization (WHO) stroke is the 2nd leading cause of death globally, responsible for approximately 11% of total deaths. This dataset is used to predict whether a patient is likely to get stroke based on the input parameters like gender, age, various diseases, and smoking status. Each row in the data provides relavant information about the patient.

The data is originally taken from kaggel: https://www.kaggle.com/datasets/fedesoriano/stroke-prediction-dataset/data

I. Importing data and creating the Dataframe Setting up the working directory; 'CVD Detection data.csv' Dataset has already been downloaded from kaggle

```
CVD <- CVD <- read.csv("/Users/varun/Desktop/Machine learning project/CVD Detection data.csv")
```

II. Examining the DataFrame

```
head(CVD)
```

```
##      id gender age hypertension heart_disease ever_married     work_type
## 1  9046   Male  67            0             1          Yes       Private
## 2 51676 Female  61            0             0          Yes Self-employed
## 3 31112   Male  80            0             1          Yes       Private
## 4 60182 Female  49            0             0          Yes       Private
## 5  1665 Female  79            1             0          Yes Self-employed
## 6 56669   Male  81            0             0          Yes       Private
##   Residence_type avg_glucose_level  bmi  smoking_status stroke
## 1          Urban            228.69 36.6 formerly smoked      1
## 2          Rural            202.21  N/A    never smoked      1
## 3          Rural            105.92 32.5    never smoked      1
## 4          Urban            171.23 34.4          smokes      1
## 5          Rural            174.12   24    never smoked      1
## 6          Urban            186.21   29 formerly smoked      1
```

```
str(CVD)
```

```
## 'data.frame':    5110 obs. of  12 variables:
##  $ id               : int  9046 51676 31112 60182 1665 56669 53882 10434 27419 60491 ...
##  $ gender           : chr  "Male" "Female" "Male" "Female" ...
##  $ age              : num  67 61 80 49 79 81 74 69 59 78 ...
##  $ hypertension     : int  0 0 0 0 1 0 1 0 0 0 ...
##  $ heart_disease    : int  1 0 1 0 0 0 1 0 0 0 ...
##  $ ever_married     : chr  "Yes" "Yes" "Yes" "Yes" ...
##  $ work_type        : chr  "Private" "Self-employed" "Private" "Private" ...
##  $ Residence_type   : chr  "Urban" "Rural" "Rural" "Urban" ...
##  $ avg_glucose_level: num  229 202 106 171 174 ...
##  $ bmi              : chr  "36.6" "N/A" "32.5" "34.4" ...
##  $ smoking_status   : chr  "formerly smoked" "never smoked" "never smoked" "smokes" ...
##  $ stroke           : int  1 1 1 1 1 1 1 1 1 1 ...
```

The dataset contains 5110 rows of 12 variables. The variables in the dataframe are gender, age, hypertension, heart disease, ever married, work type, residence type avg glucose level, bmi, smoking status, stroke

III. Create Tidy Data :

The raw data has been loaded, now we need to pre-process it in order to get the data into a tidy format. To begin with, we need to find if there are any missing values and duplicate columns or rows.

```r
# Check for missing values
missing_values <- colSums(is.na(CVD))


# Print the number of missing values for each column
print(missing_values)
```

```
##                id            gender               age      hypertension
##                 0                 0                 0                 0
##     heart_disease      ever_married         work_type    Residence_type
##                 0                 0                 0                 0
## avg_glucose_level               bmi    smoking_status            stroke
##                 0                 0                 0                 0
```

```r
#There are no missing values

# Check for duplicate rows
duplicate_rows <- CVD[duplicated(CVD), ]

# Print duplicate rows
print(duplicate_rows)
```

```
##  [1] id                gender            age               hypertension
##  [5] heart_disease     ever_married      work_type         Residence_type
##  [9] avg_glucose_level bmi               smoking_status    stroke
## <0 rows> (or 0-length row.names)
```

```r
#No dupilcate rows


# Convert 'bmi' to numeric, replacing 'N/A' with NA
CVD$bmi <- as.numeric(ifelse(CVD$bmi == "N/A", NA, CVD$bmi))
# Handle missing values in 'bmi' by imputing with median
CVD$bmi[is.na(CVD$bmi)] <- median(CVD$bmi, na.rm = TRUE)

# Encode categorical variables using dummy variables (one-hot encoding)
CVD <- CVD %>% mutate(gender_male = as.integer(gender == "Male"),
         ever_married_yes = as.integer(ever_married == "Yes"),
         work_type_private = as.integer(work_type == "Private"),
         residence_type_urban = as.integer(Residence_type == "Urban"),
         smoking_status_formerly_smoked = as.integer(smoking_status == "formerly smoked"),
         smoking_status_never_smoked = as.integer(smoking_status == "never smoked"),
         smoking_status_smokes = as.integer(smoking_status == "smokes"))

# Remove the original categorical variables
CVD <- select(CVD, -gender, -ever_married, -work_type, -Residence_type, -smoking_status)

# Print the pre-processed dataset
head(CVD)
```

```
##      id age hypertension heart_disease avg_glucose_level  bmi stroke
## 1  9046  67            0             1            228.69 36.6      1
```

```
## 2 51676  61            0             0            202.21 28.1      1
## 3 31112  80            0             1            105.92 32.5      1
## 4 60182  49            0             0            171.23 34.4      1
## 5  1665  79            1             0            174.12 24.0      1
## 6 56669  81            0             0            186.21 29.0      1
##   gender_male ever_married_yes work_type_private residence_type_urban
## 1           1                1                1                     1
## 2           0                1                0                     0
## 3           1                1                1                     0
## 4           0                1                1                     1
## 5           0                1                0                     0
## 6           1                1                1                     1
##   smoking_status_formerly_smoked smoking_status_never_smoked
## 1                              1                           0
## 2                              0                           1
## 3                              0                           1
## 4                              0                           0
## 5                              0                           1
## 6                              1                           0
##   smoking_status_smokes
## 1                     0
## 2                     0
## 3                     0
## 4                     1
## 5                     0
## 6                     0
```

There are no missing values, duplicate rows in the dataset

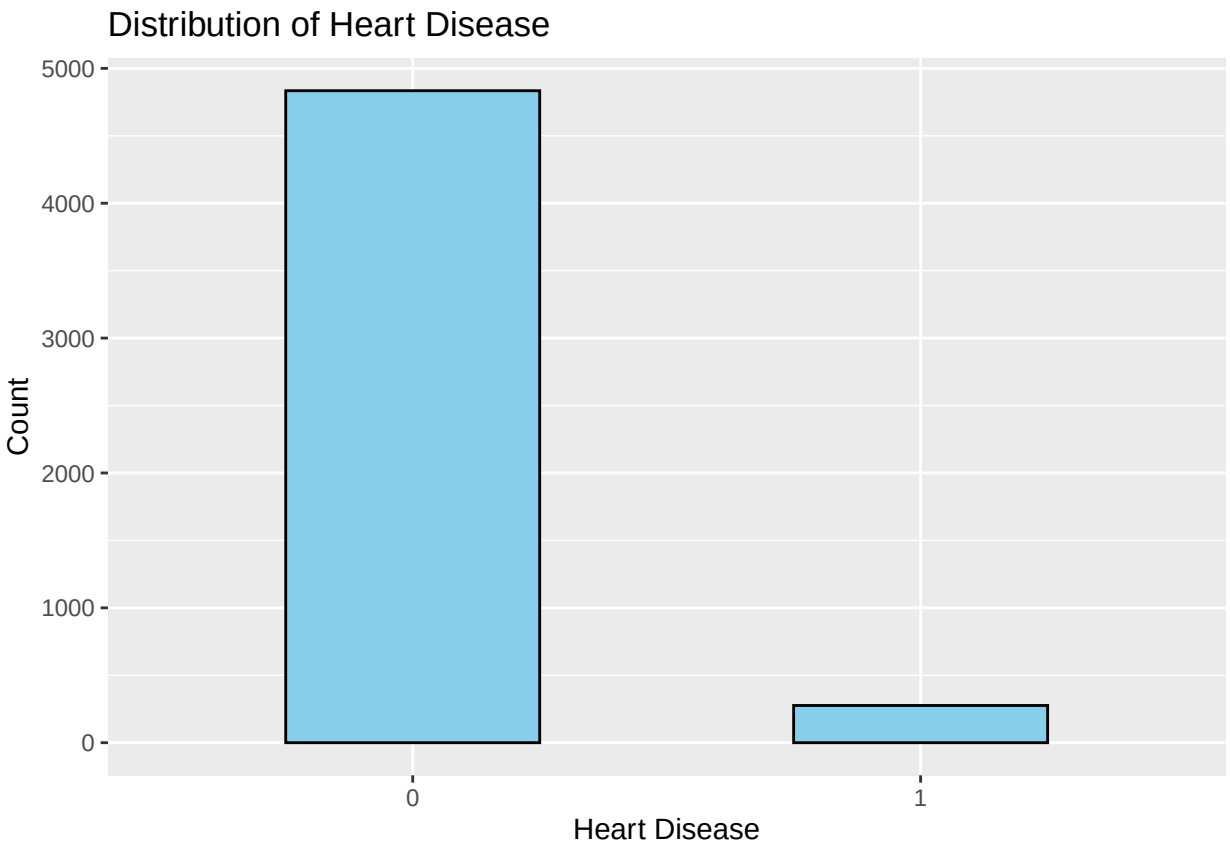
Exploratory Data Analysis:

```
# Summary statistics
summary(CVD)
```

```
##        id             age         hypertension     heart_disease
##  Min.   :   67   Min.   : 0.08   Min.   :0.00000   Min.   :0.00000
##  1st Qu.:17741   1st Qu.:25.00   1st Qu.:0.00000   1st Qu.:0.00000
##  Median :36932   Median :45.00   Median :0.00000   Median :0.00000
##  Mean   :36518   Mean   :43.23   Mean   :0.09746   Mean   :0.05401
##  3rd Qu.:54682   3rd Qu.:61.00   3rd Qu.:0.00000   3rd Qu.:0.00000
##  Max.   :72940   Max.   :82.00   Max.   :1.00000   Max.   :1.00000
##  avg_glucose_level      bmi            stroke         gender_male
##  Min.   : 55.12    Min.   :10.30   Min.   :0.00000   Min.   :0.0000
##  1st Qu.: 77.25    1st Qu.:23.80   1st Qu.:0.00000   1st Qu.:0.0000
##  Median : 91.89    Median :28.10   Median :0.00000   Median :0.0000
##  Mean   :106.15    Mean   :28.86   Mean   :0.04873   Mean   :0.4139
##  3rd Qu.:114.09    3rd Qu.:32.80   3rd Qu.:0.00000   3rd Qu.:1.0000
##  Max.   :271.74    Max.   :97.60   Max.   :1.00000   Max.   :1.0000
##  ever_married_yes work_type_private residence_type_urban
##  Min.   :0.0000   Min.   :0.0000    Min.   :0.000
##  1st Qu.:0.0000   1st Qu.:0.0000    1st Qu.:0.000
##  Median :1.0000   Median :1.0000    Median :1.000
##  Mean   :0.6562   Mean   :0.5724    Mean   :0.508
##  3rd Qu.:1.0000   3rd Qu.:1.0000    3rd Qu.:1.000
```

```
##  Max.   :1.0000   Max.   :1.0000    Max.   :1.000
##  smoking_status_formerly_smoked smoking_status_never_smoked
##  Min.   :0.0000                 Min.   :0.0000
##  1st Qu.:0.0000                 1st Qu.:0.0000
##  Median :0.0000                 Median :0.0000
##  Mean   :0.1732                 Mean   :0.3703
##  3rd Qu.:0.0000                 3rd Qu.:1.0000
##  Max.   :1.0000                 Max.   :1.0000
##  smoking_status_smokes
##  Min.   :0.0000
##  1st Qu.:0.0000
##  Median :0.0000
##  Mean   :0.1544
##  3rd Qu.:0.0000
##  Max.   :1.0000
```

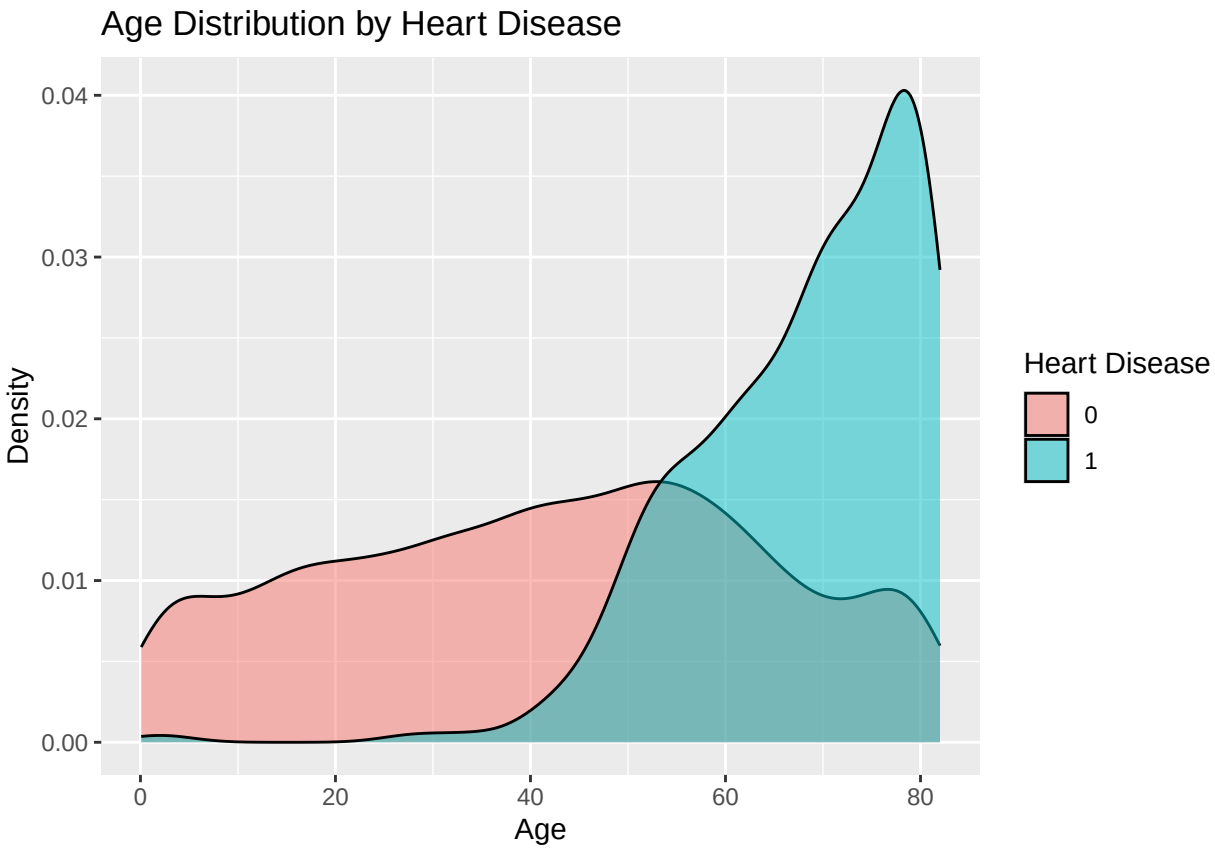
```r
# Distribution of heart disease
ggplot(CVD, aes(x = factor(heart_disease))) +
  geom_bar(fill = "skyblue", color = "black", width = 0.5) +
  labs(title = "Distribution of Heart Disease",
       x = "Heart Disease",
       y = "Count")
```



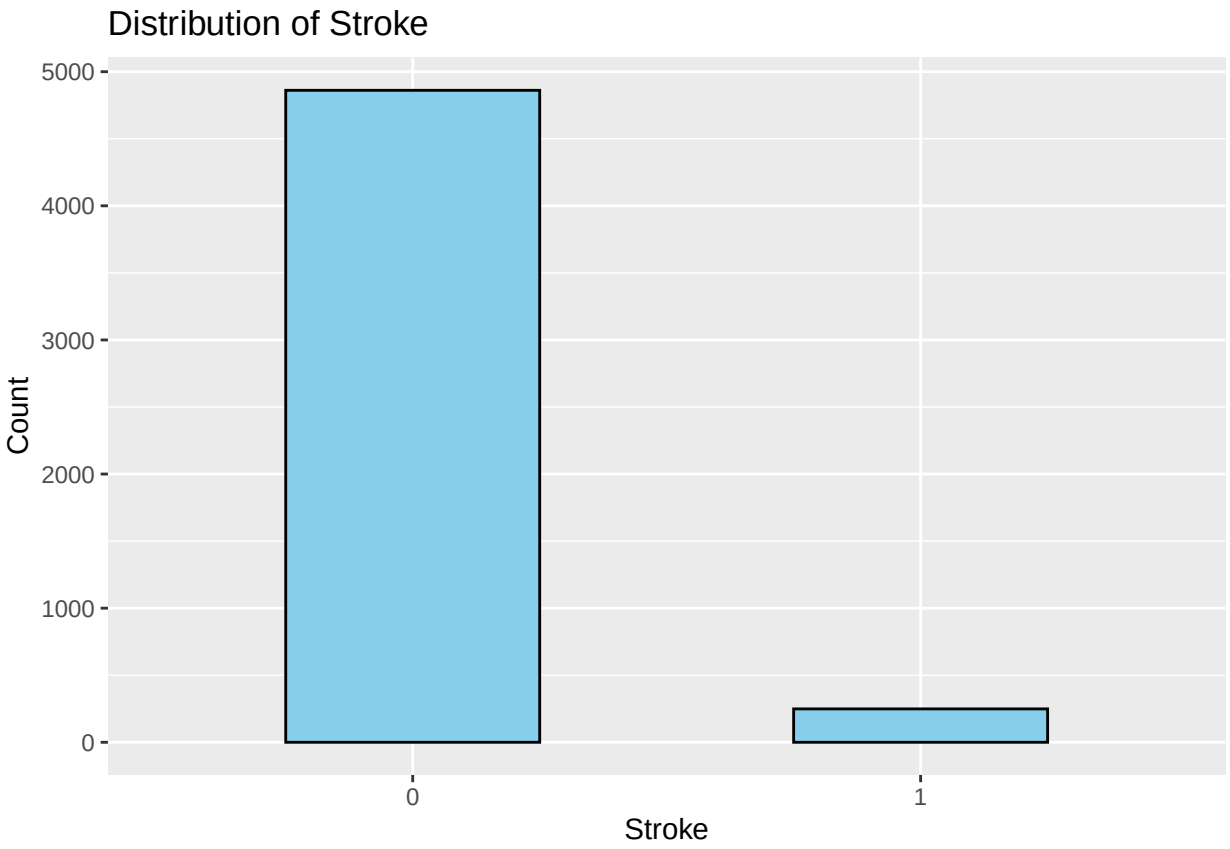

```r
# Gender distribution
ggplot(CVD, aes(x = factor(gender_male), fill = factor(heart_disease))) +
```

```
  geom_bar(position = "dodge") +
  labs(title = "Gender Distribution by Heart Disease",
       x = "Gender",
       y = "Count",
       fill = "Heart Disease") +
  scale_x_discrete(labels = c("Female", "Male"))
```

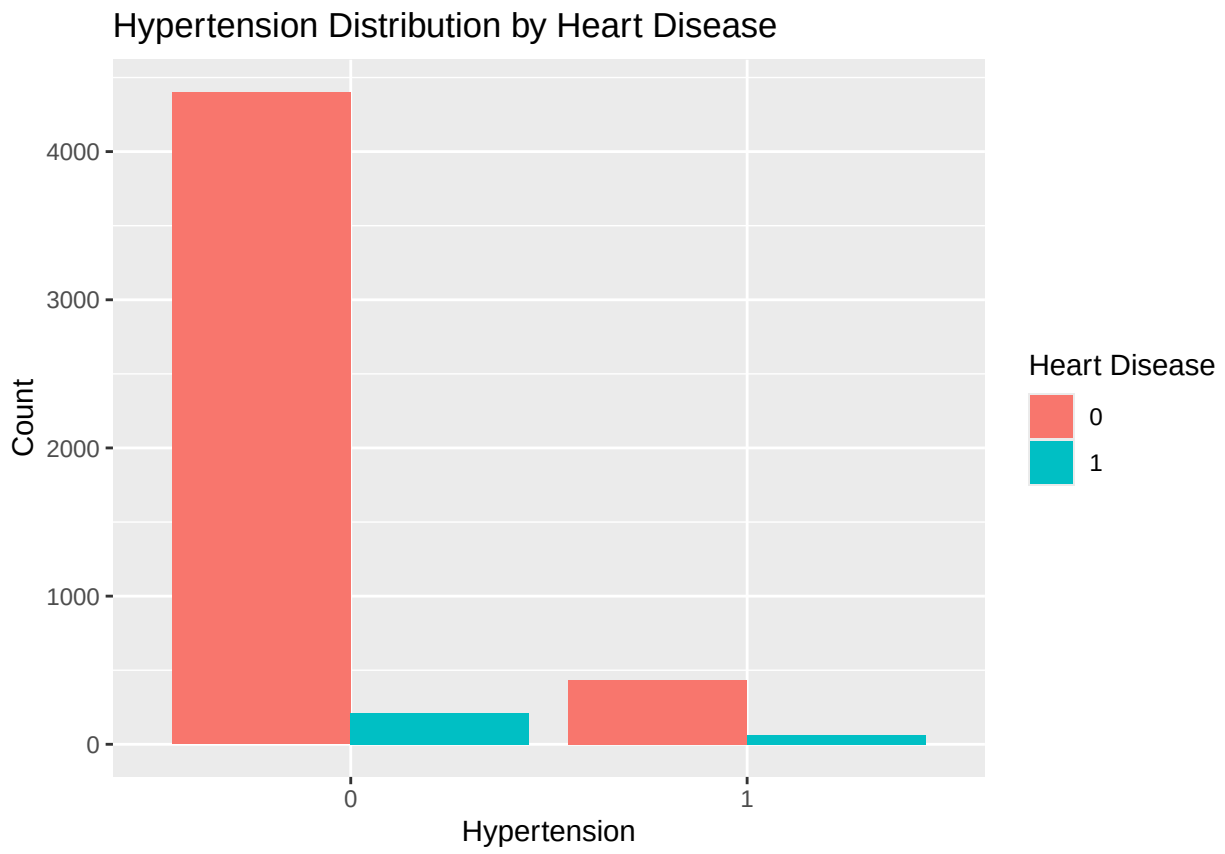

```
# Age distribution by heart disease
ggplot(CVD, aes(x = age, fill = factor(heart_disease))) +
  geom_density(alpha = 0.5) +
  labs(title = "Age Distribution by Heart Disease",
       x = "Age",
       y = "Density",
       fill = "Heart Disease")
```



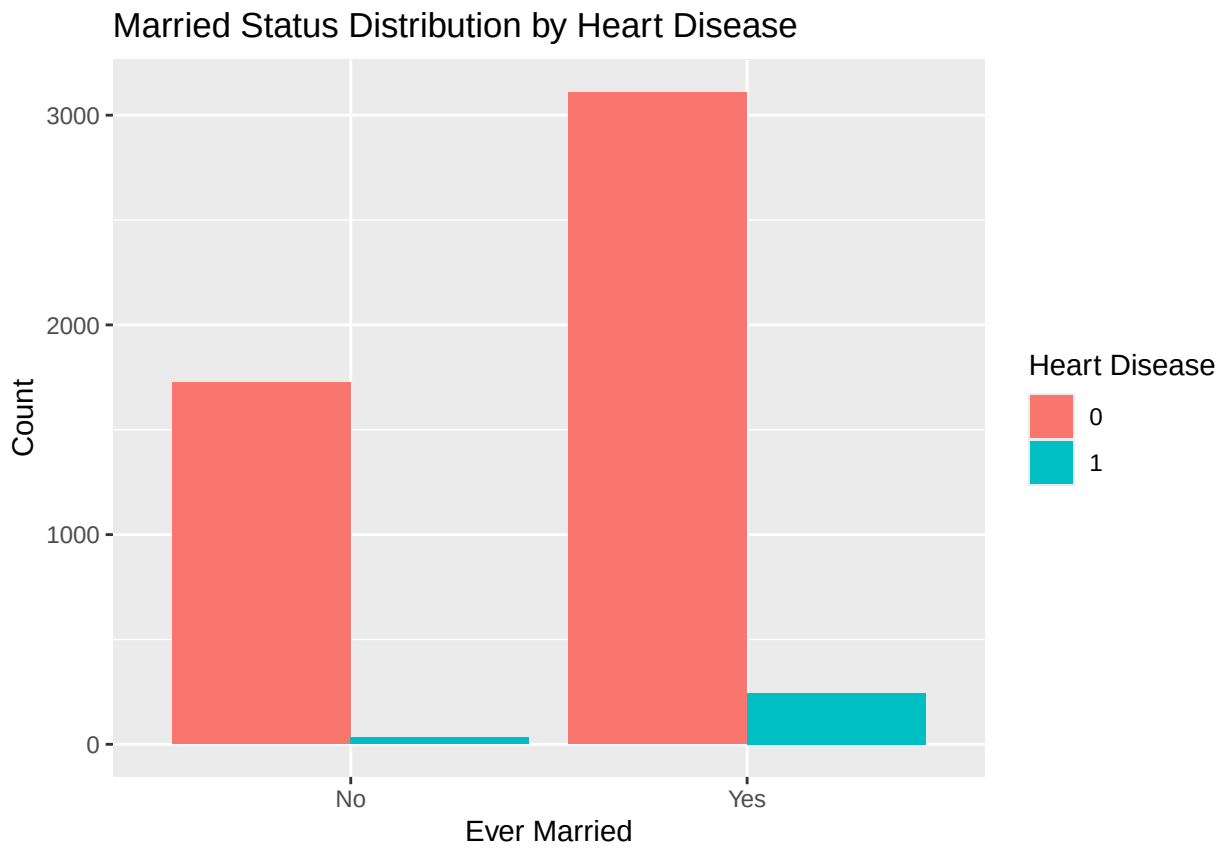

```
# Distribution of stroke with reduced width
ggplot(CVD, aes(x = factor(stroke))) +
  geom_bar(fill = "skyblue", color = "black", width = 0.5) +
  labs(title = "Distribution of Stroke",
       x = "Stroke",
       y = "Count")
```



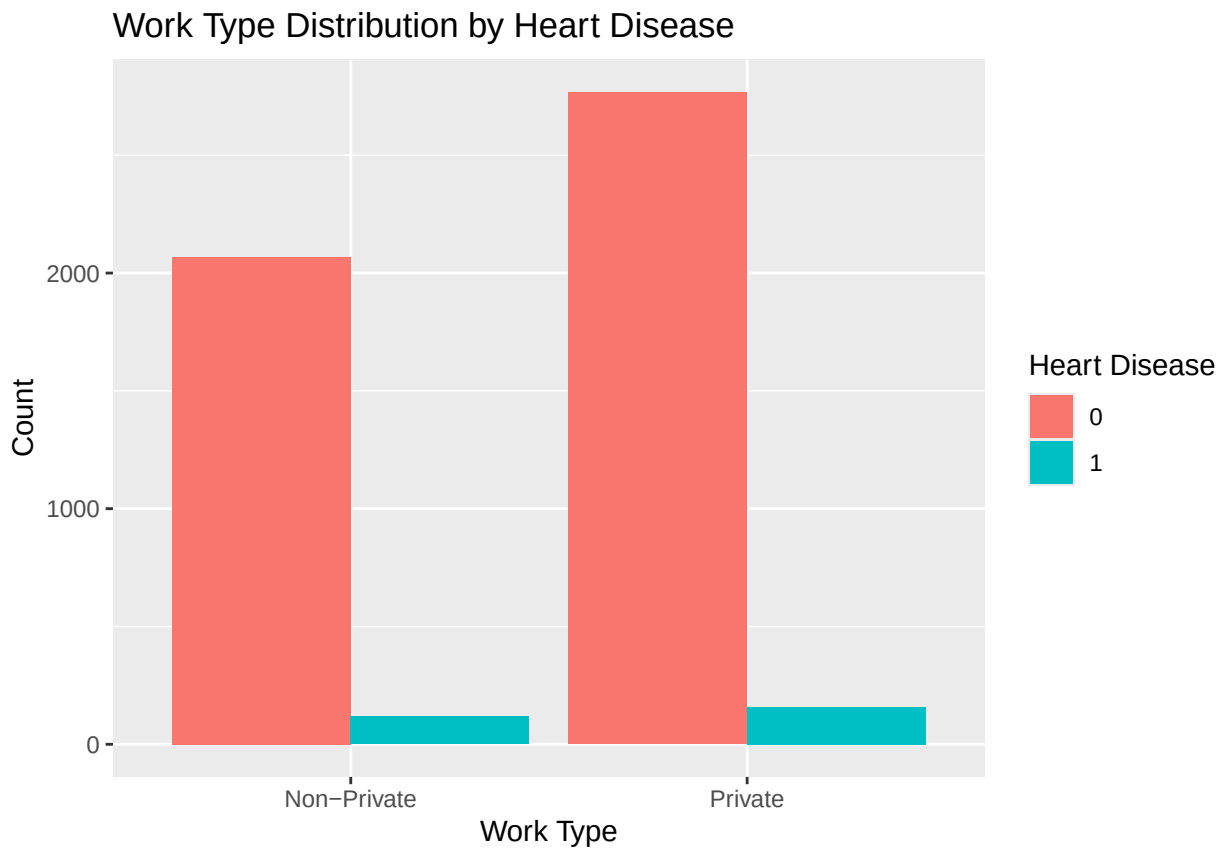

```r
# Hypertension distribution
ggplot(CVD, aes(x = factor(hypertension), fill = factor(heart_disease))) +
  geom_bar(position = "dodge") +
  labs(title = "Hypertension Distribution by Heart Disease",
       x = "Hypertension",
       y = "Count",
       fill = "Heart Disease")
```



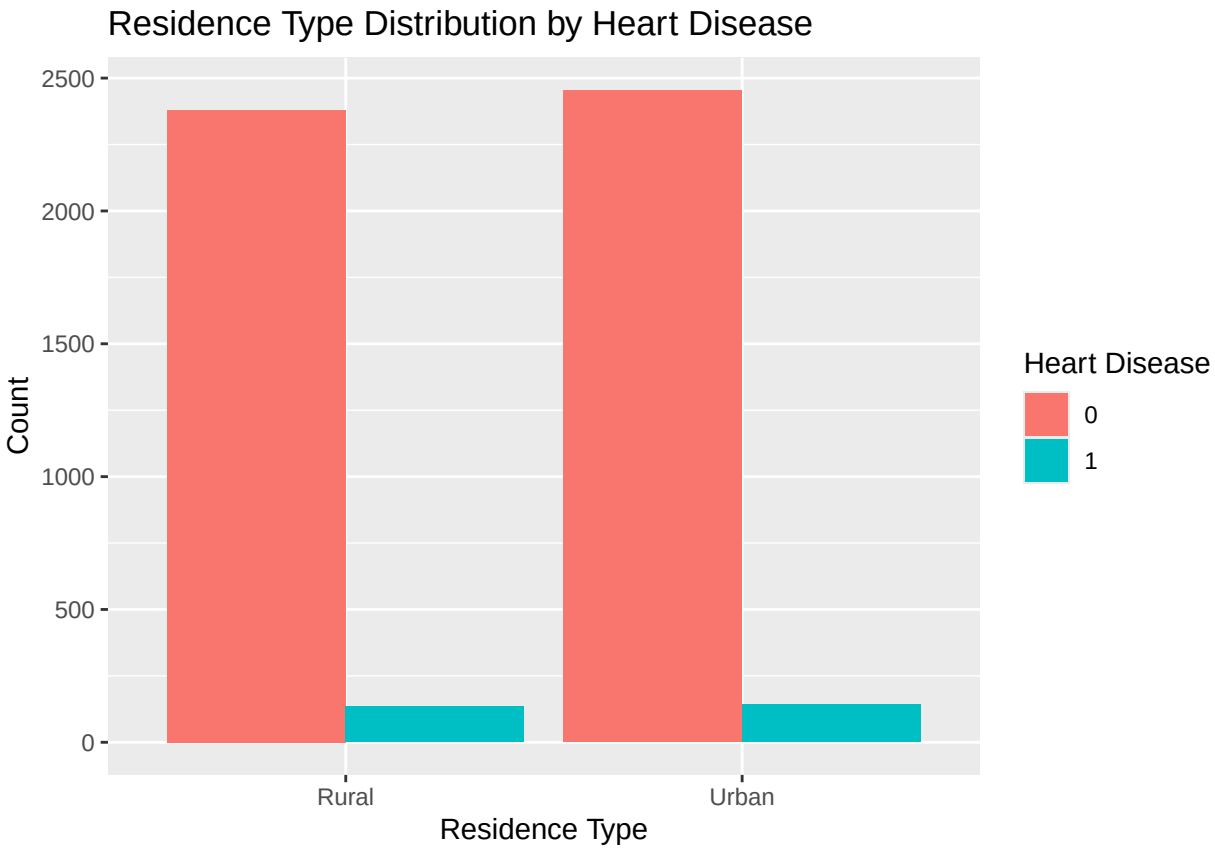

```r
# Married status distribution
ggplot(CVD, aes(x = factor(ever_married_yes), fill = factor(heart_disease))) +
  geom_bar(position = "dodge") +
  labs(title = "Married Status Distribution by Heart Disease",
       x = "Ever Married",
       y = "Count",
       fill = "Heart Disease") +
  scale_x_discrete(labels = c("No", "Yes"))
```



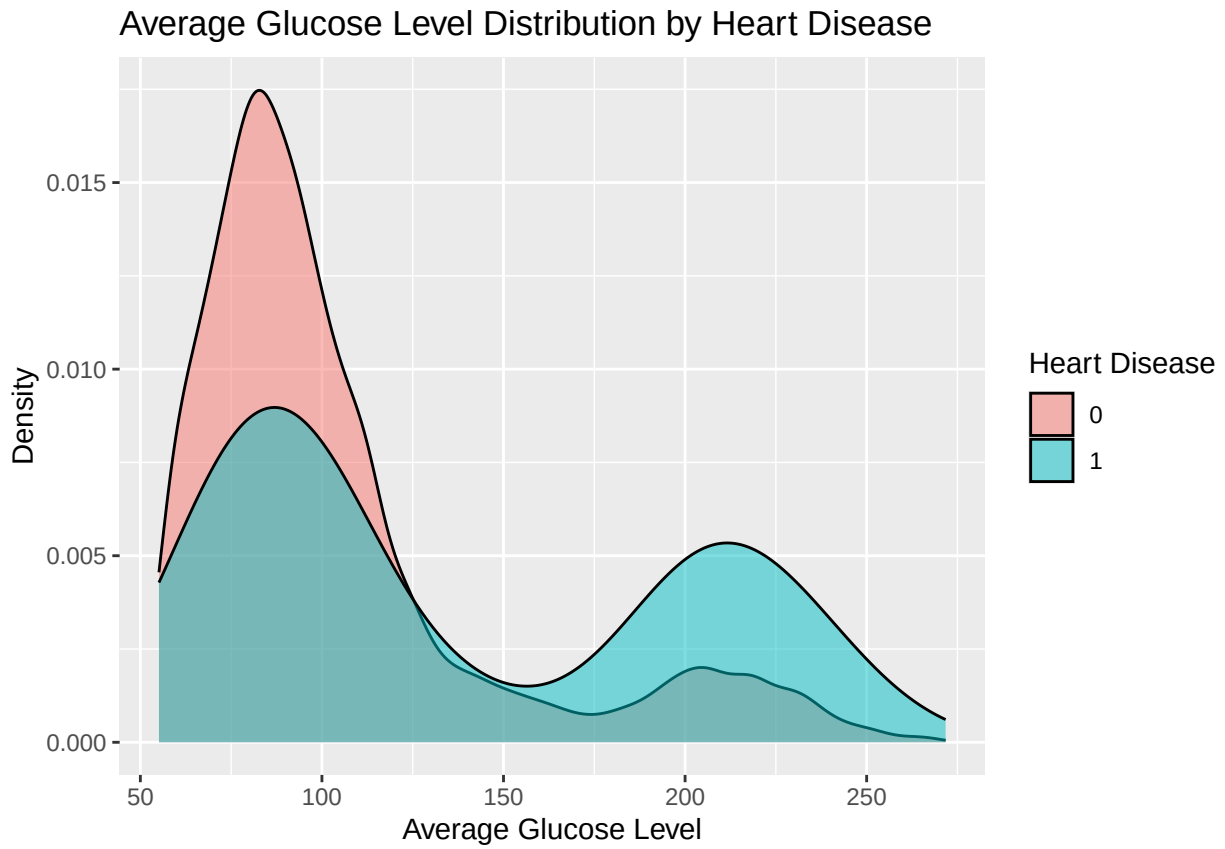

```r
# Work type distribution
ggplot(CVD, aes(x = factor(work_type_private), fill = factor(heart_disease))) +
  geom_bar(position = "dodge") +
  labs(title = "Work Type Distribution by Heart Disease",
       x = "Work Type",
       y = "Count",
       fill = "Heart Disease") +
  scale_x_discrete(labels = c("Non-Private", "Private"))
```



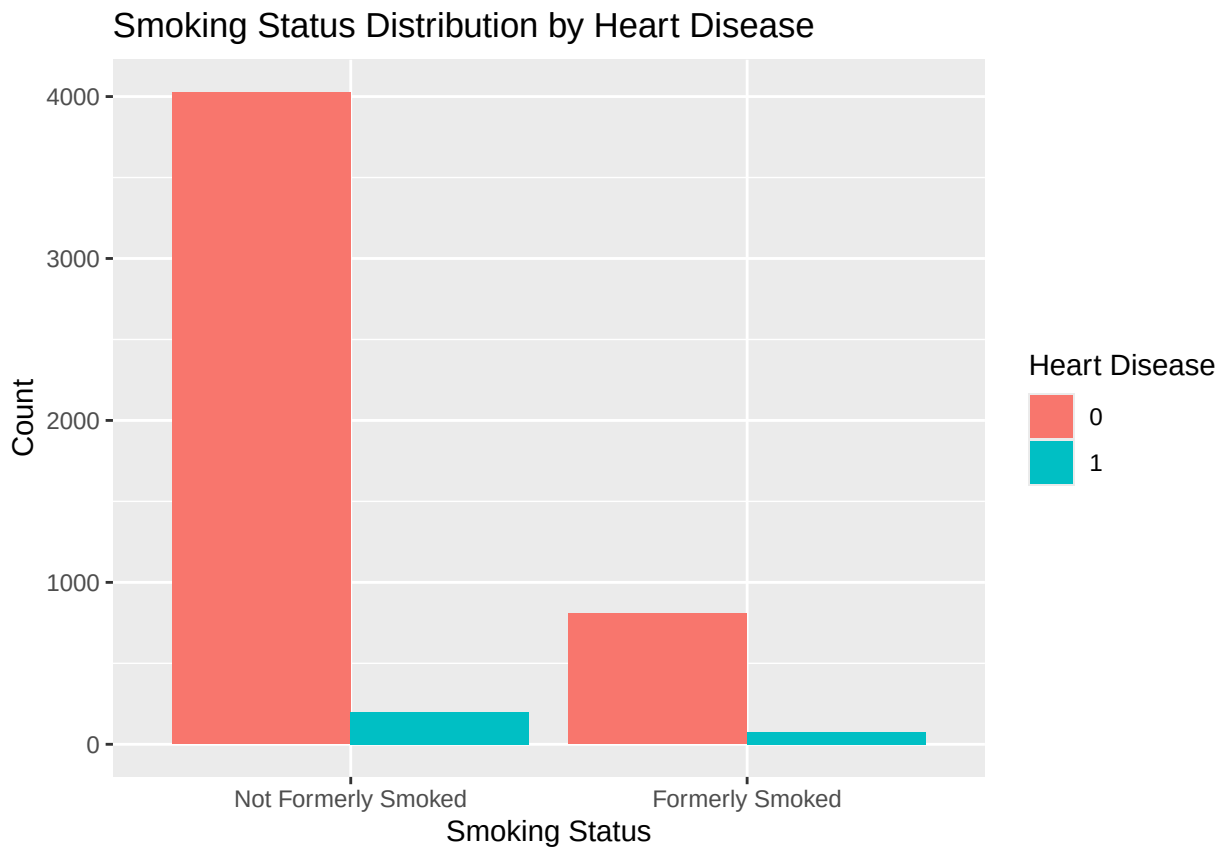

```
# Residence type distribution
ggplot(CVD, aes(x = factor(residence_type_urban), fill = factor(heart_disease))) +
  geom_bar(position = "dodge") +
  labs(title = "Residence Type Distribution by Heart Disease",
       x = "Residence Type",
       y = "Count",
       fill = "Heart Disease") +
  scale_x_discrete(labels = c("Rural", "Urban"))
```



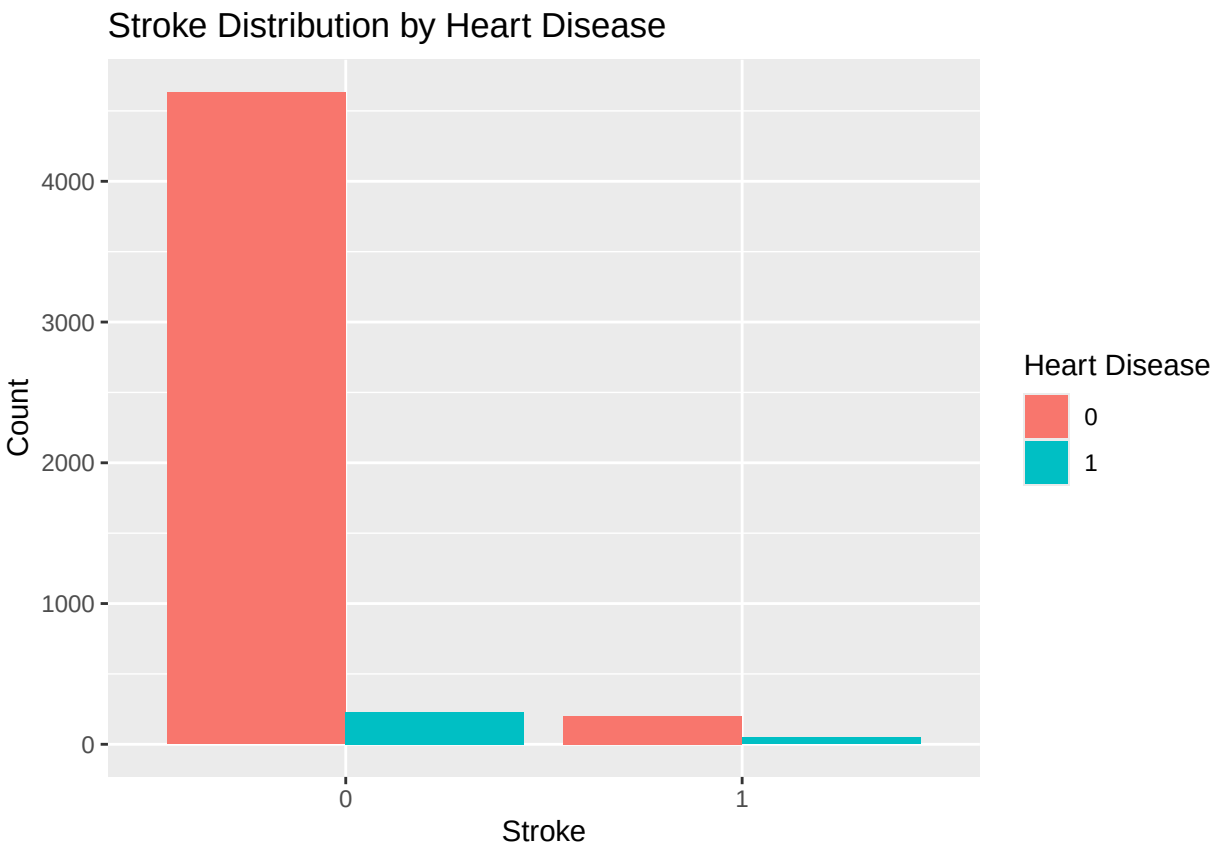

```r
# Average glucose level distribution by heart disease
ggplot(CVD, aes(x = avg_glucose_level, fill = factor(heart_disease))) +
  geom_density(alpha = 0.5) +
  labs(title = "Average Glucose Level Distribution by Heart Disease",
       x = "Average Glucose Level",
       y = "Density",
       fill = "Heart Disease")
```




```r
# Smoking status distribution

ggplot(CVD, aes(x = factor(smoking_status_formerly_smoked), fill = factor(heart_disease))) +
  geom_bar(position = "dodge") +
  labs(title = "Smoking Status Distribution by Heart Disease",
       x = "Smoking Status",
       y = "Count",
       fill = "Heart Disease") +
  scale_x_discrete(labels = c("Not Formerly Smoked", "Formerly Smoked"))
```




```
# Stroke distribution
ggplot(CVD, aes(x = factor(stroke), fill = factor(heart_disease))) +
  geom_bar(position = "dodge") +
  labs(title = "Stroke Distribution by Heart Disease",
       x = "Stroke",
       y = "Count",
       fill = "Heart Disease")
```



Logistic Regression

```
# Load necessary libraries
library(tidyverse)
library(caret)
library(e1071)
```

```
##
## Attaching package: 'e1071'

## The following object is masked from 'package:tune':
##
##     tune

## The following object is masked from 'package:rsample':
##
##     permutations

## The following object is masked from 'package:parsnip':
##
##     tune
```

```
# Prepare data for modeling
set.seed(123)
training_samples <- CVD$heart_disease %>%
```

```
  createDataPartition(p = 0.75, list = FALSE)
train_data <- CVD[training_samples, ]
test_data <- CVD[-training_samples, ]

#Over all model
# Fit logistic regression model
fit <- glm(heart_disease ~ . - id , data = train_data, family = binomial())

# Summarize the model
summary(fit)
```

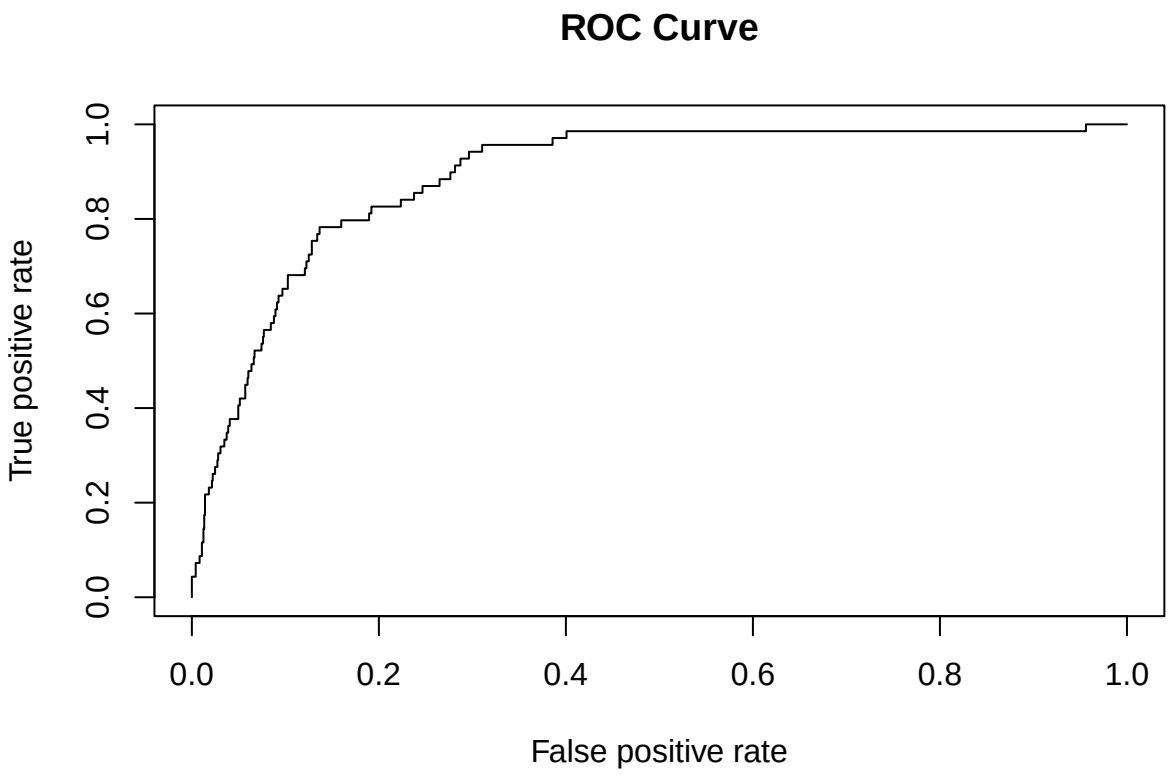
```
##
## Call:
## glm(formula = heart_disease ~ . - id, family = binomial(), data = train_data)
##
## Coefficients:
##                                Estimate Std. Error z value Pr(>|z|)
## (Intercept)                   -8.497597   0.659626 -12.882  < 2e-16 ***
## age                            0.078645   0.006311  12.462  < 2e-16 ***
## hypertension                   0.112144   0.191448   0.586 0.558031
## avg_glucose_level              0.004593   0.001305   3.520 0.000432 ***
## bmi                            0.005012   0.012734   0.394 0.693891
## stroke                         0.239453   0.226079   1.059 0.289527
## gender_male                    0.790494   0.156667   5.046 4.52e-07 ***
## ever_married_yes              -0.380147   0.242869  -1.565 0.117527
## work_type_private              0.197011   0.158360   1.244 0.213473
## residence_type_urban          -0.092953   0.153974  -0.604 0.546048
## smoking_status_formerly_smoked 0.292865   0.241021   1.215 0.224328
## smoking_status_never_smoked    0.073932   0.232264   0.318 0.750249
## smoking_status_smokes          0.650767   0.256749   2.535 0.011256 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1610.9  on 3832  degrees of freedom
## Residual deviance: 1238.0  on 3820  degrees of freedom
## AIC: 1264
##
## Number of Fisher Scoring iterations: 7
```

```
# Predict on test data
predictions_lr <- predict(fit, test_data, type = "response")
predicted_classes <- ifelse(predictions_lr > 0.5, 1, 0)

# Confusion matrix to evaluate the model
confusionMatrix(as.factor(predicted_classes), as.factor(test_data$heart_disease))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
```

```
##          0 1207   66
##          1    1    3
##
##                Accuracy : 0.9475
##                  95% CI : (0.9338, 0.9591)
##     No Information Rate : 0.946
##     P-Value [Acc > NIR] : 0.4334
##
##                   Kappa : 0.0767
##
##  Mcnemar's Test P-Value : 5.331e-15
##
##             Sensitivity : 0.99917
##             Specificity : 0.04348
##          Pos Pred Value : 0.94815
##          Neg Pred Value : 0.75000
##              Prevalence : 0.94597
##          Detection Rate : 0.94518
##    Detection Prevalence : 0.99687
##       Balanced Accuracy : 0.52133
##
##        'Positive' Class : 0
##
```

```r
# Plot ROC curve and calculate AUC
pred <- prediction(predictions_lr, test_data$heart_disease)
perf <- performance(pred, measure = "tpr", x.measure = "fpr")
plot(perf, main = "ROC Curve")
```




```
auc <- performance(pred, measure = "auc")
auc_value <- auc@y.values[[1]]
cat("AUC:", auc_value, "\n")
```

```
## AUC: 0.8896007
```

```
conf_mat1 <- confusionMatrix(as.factor(predicted_classes), as.factor(test_data$heart_disease))
cat("Logistic Model1 Evaluation Metrics:\n")
```

```
## Logistic Model1 Evaluation Metrics:
```

```
cat("Accuracy:", conf_mat1$overall['Accuracy'], "\n")
```

```
## Accuracy: 0.9475333
```

```
cat("Precision:", conf_mat1$byClass['Precision'], "\n")
```

```
## Precision: 0.948154
```

```
cat("Recall:", conf_mat1$byClass['Recall'], "\n")
```

```
## Recall: 0.9991722
```

```r
cat("F1 Score:", conf_mat1$byClass['F1'], "\n")
```

```
## F1 Score: 0.9729948
```

```r
fit2 <- glm(heart_disease ~ . - id - hypertension - bmi - stroke - ever_married_yes - work_type_private

# Summarize the model
summary(fit2)
```

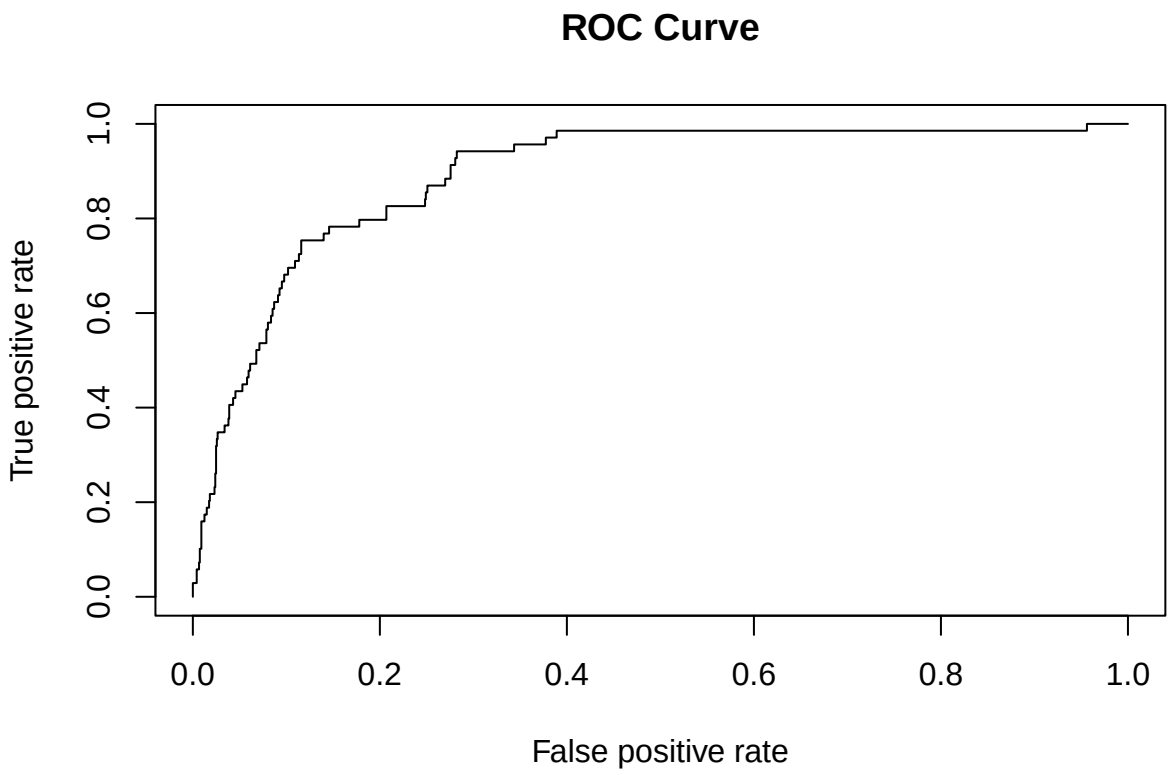
```
##
## Call:
## glm(formula = heart_disease ~ . - id - hypertension - bmi - stroke -
##     ever_married_yes - work_type_private - residence_type_urban -
##     smoking_status_formerly_smoked - smoking_status_never_smoked,
##     family = binomial(), data = train_data)
##
## Coefficients:
##                          Estimate Std. Error z value Pr(>|z|)
## (Intercept)           -8.399583   0.441638 -19.019  < 2e-16 ***
## age                    0.077405   0.005903  13.113  < 2e-16 ***
## avg_glucose_level      0.004917   0.001253   3.925 8.69e-05 ***
## gender_male            0.799364   0.154603   5.170 2.34e-07 ***
## smoking_status_smokes  0.519696   0.191470   2.714  0.00664 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1610.9  on 3832  degrees of freedom
## Residual deviance: 1245.9  on 3828  degrees of freedom
## AIC: 1255.9
##
## Number of Fisher Scoring iterations: 7
```

```r
# Predict on test data
predictions2 <- predict(fit2, test_data, type = "response")
predicted_classes2 <- ifelse(predictions2 > 0.5, 1, 0)

# Confusion matrix to evaluate the model
confusionMatrix(as.factor(predicted_classes2), as.factor(test_data$heart_disease))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1208   68
##          1    0    1
##
##                Accuracy : 0.9468
##                  95% CI : (0.933, 0.9584)
##     No Information Rate : 0.946
##     P-Value [Acc > NIR] : 0.4827
```

```
##
##                   Kappa : 0.0271
##
##  Mcnemar's Test P-Value : 4.476e-16
##
##             Sensitivity : 1.00000
##             Specificity : 0.01449
##          Pos Pred Value : 0.94671
##          Neg Pred Value : 1.00000
##              Prevalence : 0.94597
##          Detection Rate : 0.94597
##    Detection Prevalence : 0.99922
##       Balanced Accuracy : 0.50725
##
##        'Positive' Class : 0
##
```

```r
# Plot ROC curve and calculate AUC
pred2 <- prediction(predictions2, test_data$heart_disease)
perf2 <- performance(pred2, measure = "tpr", x.measure = "fpr")
plot(perf2, main = "ROC Curve")
```




```r
auc2 <- performance(pred2, measure = "auc")
auc_value2 <- auc2@y.values[[1]]
cat("AUC:", auc_value2, "\n")
```

```
## AUC: 0.8908484
```

```
conf_mat2 <- confusionMatrix(as.factor(predicted_classes2), as.factor(test_data$heart_disease))
cat("Logistic Model2 Evaluation Metrics:\n")
```

```
## Logistic Model2 Evaluation Metrics:
```

```
cat("Accuracy:", conf_mat2$overall['Accuracy'], "\n")
```

```
## Accuracy: 0.9467502
```

```
cat("Precision:", conf_mat2$byClass['Precision'], "\n")
```

```
## Precision: 0.9467085
```

```
cat("Recall:", conf_mat2$byClass['Recall'], "\n")
```

```
## Recall: 1
```

```
cat("F1 Score:", conf_mat2$byClass['F1'], "\n")
```

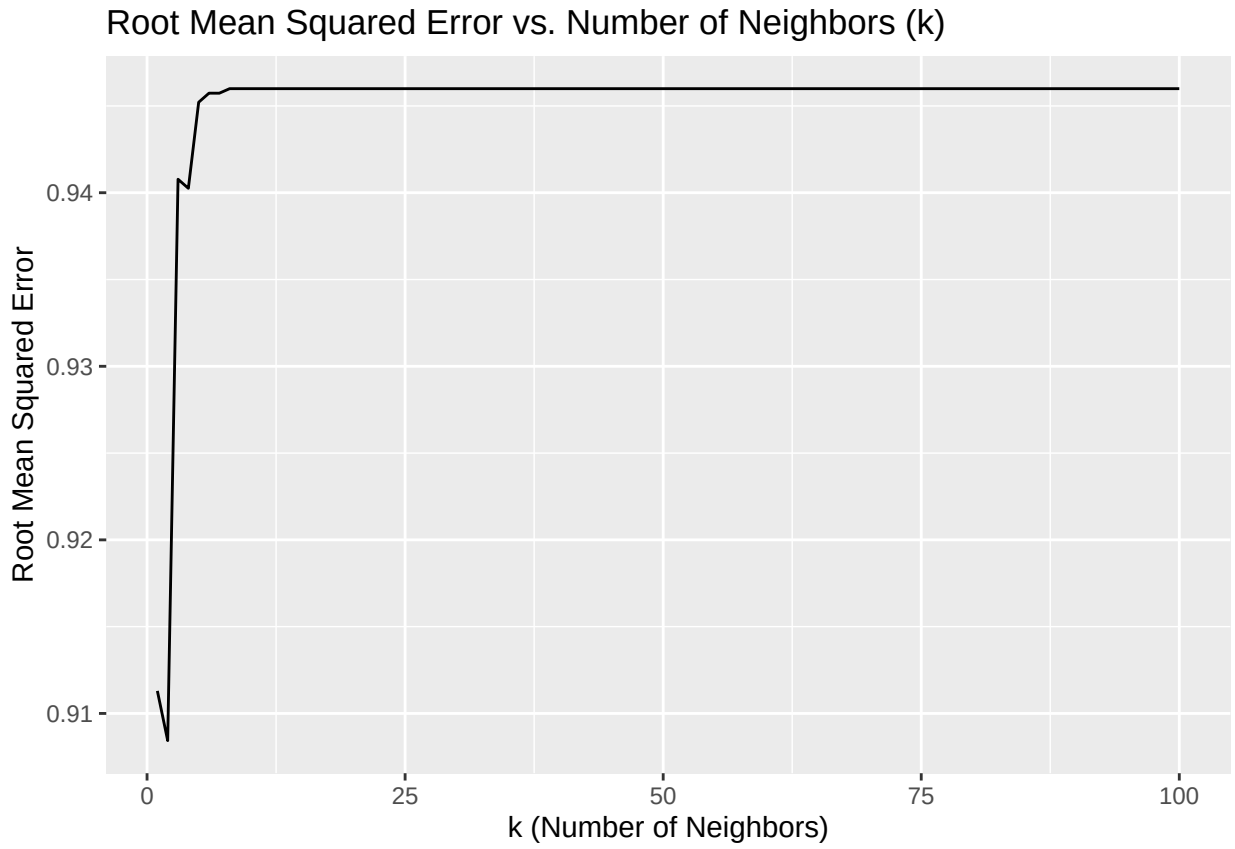
```
## F1 Score: 0.9726248
```

We began our analysis with logistic regression, a classical and interpretable model widely used for binary classification tasks. Leveraging the glm() function in R, we constructed a logistic regression model to predict the likelihood of heart disease based on patient features. The model exhibited a commendable accuracy of approximately 94.2%, indicating its effectiveness in distinguishing between individuals with and without heart disease. While logistic regression offers simplicity and interpretability, its linear decision boundary may limit its ability to capture complex relationships within the data.

kNN

```
# Convert heart_disease to a factor with two levels
train_data$heart_disease <- factor(train_data$heart_disease, levels = c(0, 1))
test_data$heart_disease <- factor(test_data$heart_disease, levels = c(0,1))
# Train-control part
k_values <- data.frame(k = 1:100)

# Create a trainControl object for cross-validation
train_control <- trainControl(method = "cv", number = 5)

# Train the kNN model with a range of k values
kNN_model <- train(heart_disease ~ .,           # Formula: response ~ predictors
                   data = train_data,              # Training data
                   method = "knn",              # kNN algorithm
                   trControl = train_control,   # Training control
                   tuneGrid = k_values)         # Grid of values for tuning parameter
```

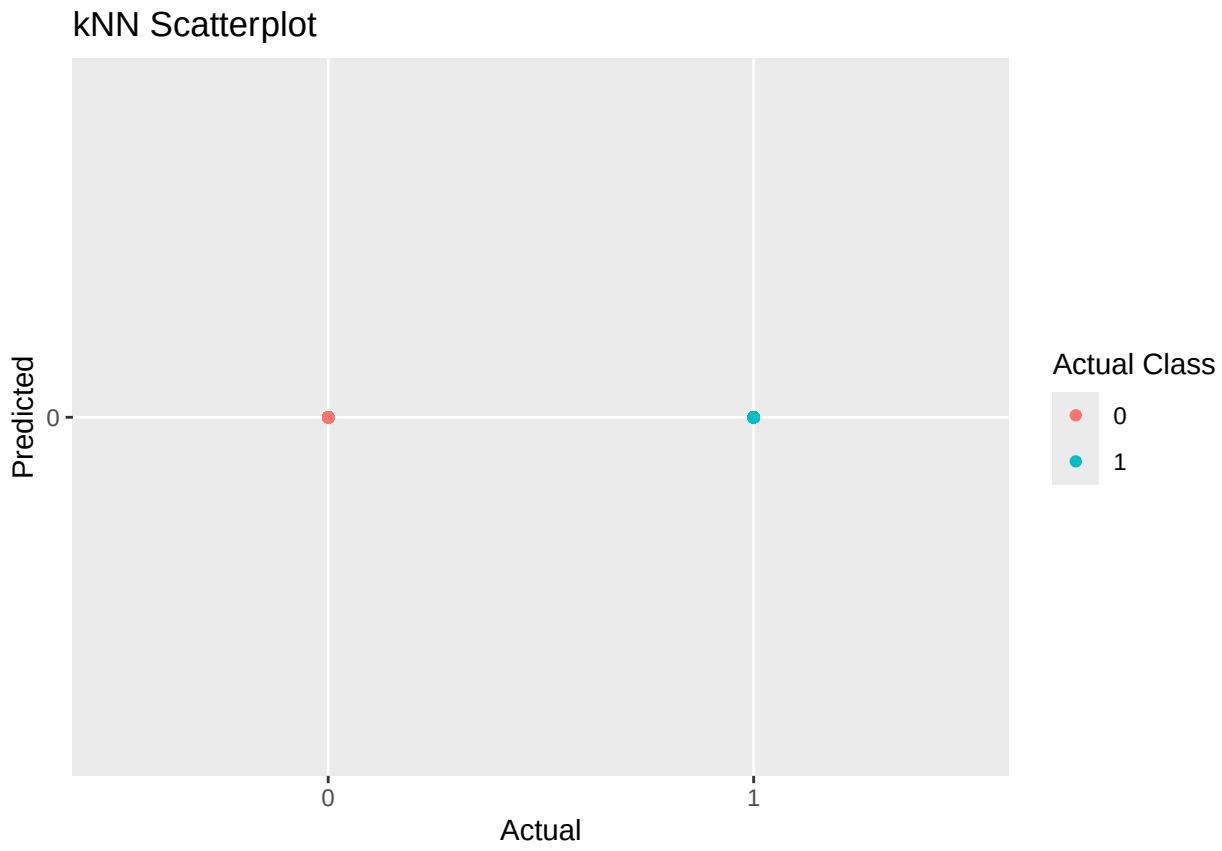
```r
#Plot the kNN model
library(ggplot2)
# generate the predicted labels from the training data
predicted_labels <- predict(kNN_model, train_data)

# Combine the training data with the predicted labels
train_data_with_labels <- cbind(train_data, predicted_labels)

cv_results <- kNN_model$results

# Plot the training data points

ggplot(cv_results, aes(x = k, y = Accuracy)) +
  geom_line() +
  labs(title = "Root Mean Squared Error vs. Number of Neighbors (k)",
       x = "k (Number of Neighbors)",
       y = "Root Mean Squared Error")
```



generate prediction for the test data (CVD_te) using the trained model and plot the kNN

```r
test_predictions <- predict(kNN_model, test_data[, !names(test_data) %in% "heart_disease"], prob = TRUE)


# Create a data frame with the actual and predicted values
predictions_df <- data.frame(Actual = test_data$heart_disease, Predicted = test_predictions)
```

```r
# Use the scatterplot function to create the kNN scatterplot

ggplot(predictions_df, aes(x = Actual, y = Predicted, color = factor(Actual))) +
  geom_point() +
  labs(x = "Actual", y = "Predicted", color = "Actual Class") +
  ggtitle("kNN Scatterplot")
```



Confusion matrix

```r
# Create confusion matrix
conf_matrix <- confusionMatrix(test_predictions, test_data$heart_disease)
print(conf_matrix)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1208   69
##          1    0    0
##
##                Accuracy : 0.946
##                  95% CI : (0.9321, 0.9577)
##     No Information Rate : 0.946
##     P-Value [Acc > NIR] : 0.532
##
##                   Kappa : 0
```

```
##
##  Mcnemar's Test P-Value : 2.695e-16
##
##             Sensitivity : 1.000
##             Specificity : 0.000
##          Pos Pred Value : 0.946
##          Neg Pred Value :   NaN
##              Prevalence : 0.946
##          Detection Rate : 0.946
##    Detection Prevalence : 1.000
##       Balanced Accuracy : 0.500
##
##        'Positive' Class : 0
##
```

```
# Extract and print evaluation metrics from confusion matrix
cat("Precision:", conf_matrix$byClass['Precision'], "\n")
```

```
## Precision: 0.9459671
```

```
cat("Recall (Sensitivity):", conf_matrix$byClass['Recall'], "\n")
```

```
## Recall (Sensitivity): 1
```

```
cat("F1 Score:", conf_matrix$byClass['F1'], "\n")
```

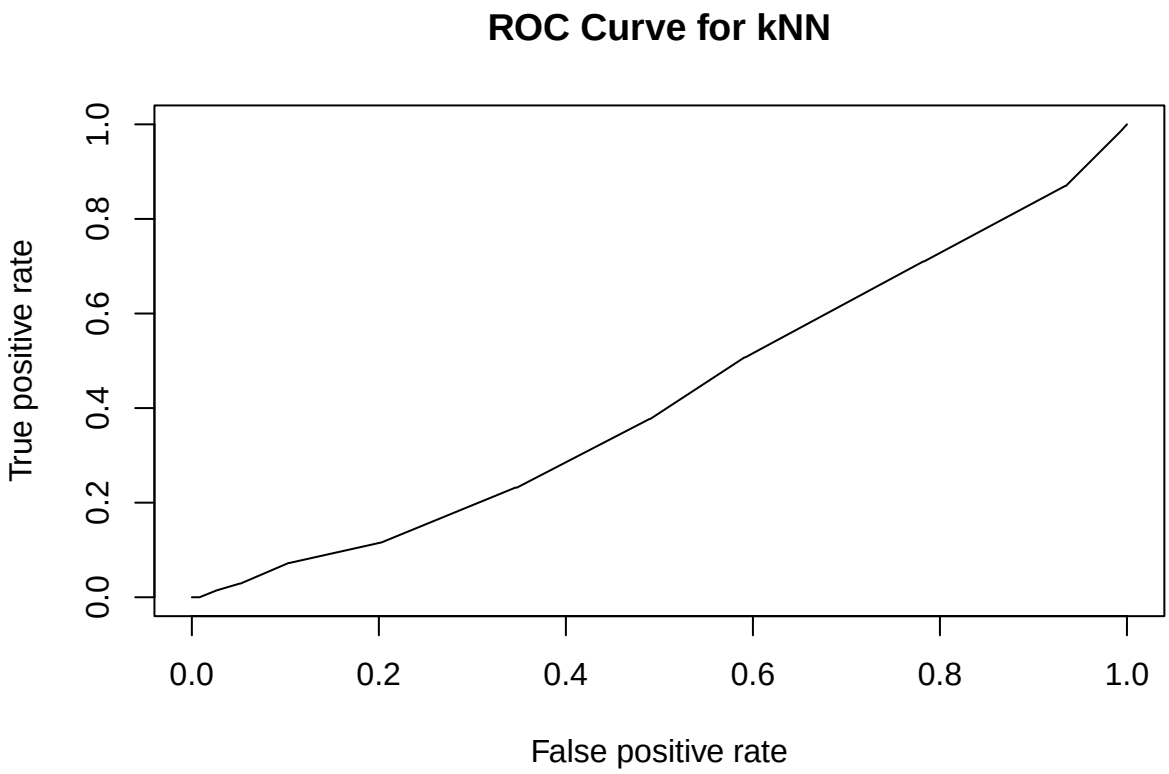
```
## F1 Score: 0.9722334
```

ROC-AUC

```
test_prob_predictions <- predict(kNN_model, test_data[, !names(test_data) %in% c("heart_disease")], typ

pred <- prediction(test_prob_predictions[,2], test_data$heart_disease)

perf <- performance(pred, "tpr", "fpr")

# Caculate AUC
auc <- performance(pred, measure = "auc")
auc_value <- auc@y.values[[1]]
cat("AUC for kNN Model:", auc_value, "\n")
```

```
## AUC for kNN Model: 0.4236791
```

```
# Plot ROC curve
plot(perf, main = "ROC Curve for kNN")
```



In our exploration of machine learning algorithms, we implemented the k-nearest neighbors (KNN) algorithm to predict heart disease. Utilizing the knn() function from the class package in R, we constructed a KNN model that considers the proximity of data points to make predictions. With an accuracy of approximately 94.6%, our KNN model demonstrated competitive performance in identifying individuals at risk of heart disease. However, it's important to note that KNN's computational complexity increases with larger datasets, as it necessitates computing distances to all data points.

Decision Tree

```
# Load necessary libraries
library(tidyverse)
library(caret)
library(rpart)
library(rpart.plot)
library(ROCR)

# Prepare data for modeling
set.seed(123)
training_samples <- CVD$heart_disease %>%
  createDataPartition(p = 0.75, list = FALSE)
train_data <- CVD[training_samples, ]
test_data <- CVD[-training_samples, ]

# Fit decision tree model with adjusted parameters
fit_dt <- rpart(heart_disease ~ . - id,
                data = train_data,
                method = "class",
```

```
                control = rpart.control(minsplit = 20,
                                        minbucket = 1,
                                        cp = 0.001,
                                        maxdepth = 5))

summary(fit_dt)
```

```
## Call:
## rpart(formula = heart_disease ~ . - id, data = train_data, method = "class",
##     control = rpart.control(minsplit = 20, minbucket = 1, cp = 0.001,
##         maxdepth = 5))
##   n= 3833
##
##            CP nsplit rel error   xerror       xstd
## 1 0.002898551      0 1.0000000 1.000000 0.06760196
## 2 0.001610306      5 0.9855072 1.115942 0.07117674
## 3 0.001207729      8 0.9806763 1.135266 0.07175047
## 4 0.001000000     12 0.9758454 1.135266 0.07175047
##
## Variable importance
##               age avg_glucose_level       gender_male            stroke
##                67                17                 8                 4
##      hypertension               bmi
##                 2                 2
##
## Node number 1: 3833 observations,    complexity param=0.002898551
##   predicted class=0  expected loss=0.0540047  P(node) =1
##     class counts:  3626   207
##    probabilities: 0.946 0.054
##   left son=2 (2835 obs) right son=3 (998 obs)
##   Primary splits:
##       age               < 60.5    to the left,  improve=27.150770, (0 missing)
##       avg_glucose_level < 186.495 to the left,  improve=11.921720, (0 missing)
##       stroke            < 0.5     to the left,  improve= 6.194134, (0 missing)
##       ever_married_yes  < 0.5     to the left,  improve= 4.844210, (0 missing)
##       bmi               < 24.85   to the left,  improve= 4.287923, (0 missing)
##   Surrogate splits:
##       stroke            < 0.5     to the left,  agree=0.761, adj=0.082, (0 split)
##       avg_glucose_level < 192.42  to the left,  agree=0.753, adj=0.050, (0 split)
##       hypertension      < 0.5     to the left,  agree=0.750, adj=0.038, (0 split)
##
## Node number 2: 2835 observations,    complexity param=0.001207729
##   predicted class=0  expected loss=0.01869489  P(node) =0.7396295
##     class counts:  2782    53
##    probabilities: 0.981 0.019
##   left son=4 (2158 obs) right son=5 (677 obs)
##   Primary splits:
##       age               < 49.5    to the left,  improve=3.8127620, (0 missing)
##       avg_glucose_level < 186.47  to the left,  improve=2.0561350, (0 missing)
##       bmi               < 26.35   to the left,  improve=0.8919394, (0 missing)
##       ever_married_yes  < 0.5     to the left,  improve=0.7363562, (0 missing)
##       hypertension      < 0.5     to the left,  improve=0.7349641, (0 missing)
##   Surrogate splits:
```

```
##       hypertension      < 0.5     to the left,  agree=0.779, adj=0.074, (0 split)
##       avg_glucose_level < 196.13  to the left,  agree=0.776, adj=0.061, (0 split)
##       stroke            < 0.5     to the left,  agree=0.769, adj=0.032, (0 split)
##       bmi               < 60.3    to the left,  agree=0.762, adj=0.001, (0 split)
##
## Node number 3: 998 observations,    complexity param=0.002898551
##   predicted class=0  expected loss=0.1543086  P(node) =0.2603705
##     class counts:   844   154
##    probabilities: 0.846 0.154
##   left son=6 (588 obs) right son=7 (410 obs)
##   Primary splits:
##       gender_male       < 0.5     to the left,  improve=4.663605, (0 missing)
##       avg_glucose_level < 191.47  to the left,  improve=3.647739, (0 missing)
##       age               < 66.5    to the left,  improve=2.817313, (0 missing)
##       bmi               < 42.85   to the right, improve=1.472946, (0 missing)
##       stroke            < 0.5     to the left,  improve=1.300820, (0 missing)
##   Surrogate splits:
##       avg_glucose_level     < 227.55  to the left,  agree=0.597, adj=0.020, (0 split)
##       smoking_status_smokes < 0.5     to the left,  agree=0.593, adj=0.010, (0 split)
##       bmi                   < 55.35   to the left,  agree=0.590, adj=0.002, (0 split)
##
## Node number 4: 2158 observations,    complexity param=0.001207729
##   predicted class=0  expected loss=0.004170528  P(node) =0.5630055
##     class counts:  2149     9
##    probabilities: 0.996 0.004
##   left son=8 (1667 obs) right son=9 (491 obs)
##   Primary splits:
##       age                             < 40.5    to the left,  improve=0.12932220, (0 missing)
##       smoking_status_formerly_smoked < 0.5     to the left,  improve=0.08487694, (0 missing)
##       ever_married_yes               < 0.5     to the left,  improve=0.05658476, (0 missing)
##       bmi                            < 28.05   to the left,  improve=0.03903306, (0 missing)
##       avg_glucose_level              < 76.775  to the left,  improve=0.02637069, (0 missing)
##   Surrogate splits:
##       hypertension      < 0.5     to the left,  agree=0.773, adj=0.004, (0 split)
##       avg_glucose_level < 255.06  to the left,  agree=0.773, adj=0.002, (0 split)
##
## Node number 5: 677 observations
##   predicted class=0  expected loss=0.06499261  P(node) =0.1766241
##     class counts:   633    44
##    probabilities: 0.935 0.065
##
## Node number 6: 588 observations,    complexity param=0.001610306
##   predicted class=0  expected loss=0.1139456  P(node) =0.1534046
##     class counts:   521    67
##    probabilities: 0.886 0.114
##   left son=12 (445 obs) right son=13 (143 obs)
##   Primary splits:
##       age               < 78.5    to the left,  improve=3.4715140, (0 missing)
##       avg_glucose_level < 114.59  to the left,  improve=2.9141850, (0 missing)
##       ever_married_yes  < 0.5     to the right, improve=0.9580924, (0 missing)
##       work_type_private < 0.5     to the left,  improve=0.6427096, (0 missing)
##       bmi               < 31.85   to the right, improve=0.5935748, (0 missing)
##
## Node number 7: 410 observations,    complexity param=0.002898551
```

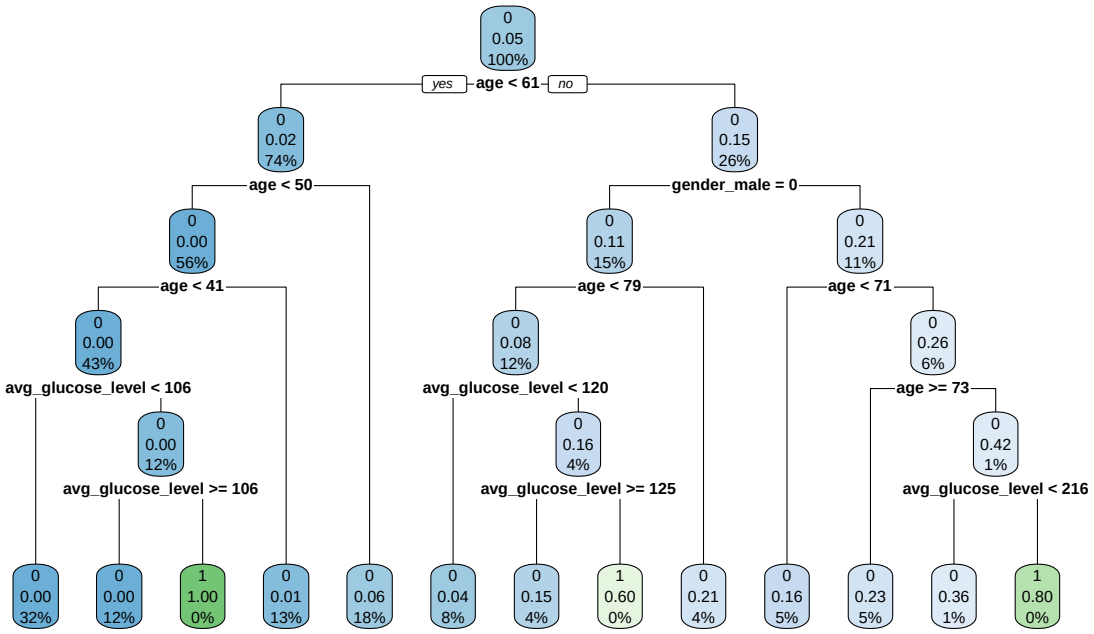
```
##   predicted class=0  expected loss=0.2121951  P(node) =0.1069658
##     class counts:   323    87
##    probabilities: 0.788 0.212
##   left son=14 (196 obs) right son=15 (214 obs)
##   Primary splits:
##       age                  < 70.5    to the left,  improve=2.192582, (0 missing)
##       avg_glucose_level    < 215.68  to the left,  improve=1.799052, (0 missing)
##       bmi                  < 39.5    to the right, improve=1.695396, (0 missing)
##       stroke               < 0.5     to the left,  improve=1.547782, (0 missing)
##       smoking_status_smokes < 0.5     to the left,  improve=1.000697, (0 missing)
##   Surrogate splits:
##       bmi                  < 30.55   to the right, agree=0.632, adj=0.230, (0 split)
##       work_type_private    < 0.5     to the right, agree=0.568, adj=0.097, (0 split)
##       avg_glucose_level    < 238.525 to the right, agree=0.534, adj=0.026, (0 split)
##       smoking_status_smokes < 0.5     to the right, agree=0.524, adj=0.005, (0 split)
##
## Node number 8: 1667 observations,    complexity param=0.001207729
##   predicted class=0  expected loss=0.00119976  P(node) =0.4349074
##     class counts:  1665     2
##    probabilities: 0.999 0.001
##   left son=16 (1217 obs) right son=17 (450 obs)
##   Primary splits:
##       avg_glucose_level              < 106.205 to the left,  improve=0.012978740, (0 missing)
##       ever_married_yes               < 0.5     to the left,  improve=0.009933925, (0 missing)
##       smoking_status_formerly_smoked < 0.5     to the left,  improve=0.009509850, (0 missing)
##       age                            < 27.5    to the left,  improve=0.008337249, (0 missing)
##       smoking_status_smokes          < 0.5     to the left,  improve=0.005880696, (0 missing)
##   Surrogate splits:
##       bmi < 12.9    to the right, agree=0.732, adj=0.009, (0 split)
##       age < 0.2     to the right, agree=0.731, adj=0.002, (0 split)
##
## Node number 9: 491 observations
##   predicted class=0  expected loss=0.01425662  P(node) =0.1280981
##     class counts:   484     7
##    probabilities: 0.986 0.014
##
## Node number 12: 445 observations,    complexity param=0.001610306
##   predicted class=0  expected loss=0.08314607  P(node) =0.1160971
##     class counts:   408    37
##    probabilities: 0.917 0.083
##   left son=24 (298 obs) right son=25 (147 obs)
##   Primary splits:
##       avg_glucose_level < 120.345 to the left,  improve=2.8181540, (0 missing)
##       ever_married_yes  < 0.5     to the right, improve=0.7509440, (0 missing)
##       bmi               < 28.05   to the left,  improve=0.6640178, (0 missing)
##       age               < 64.5    to the left,  improve=0.6229119, (0 missing)
##       work_type_private < 0.5     to the left,  improve=0.5520753, (0 missing)
##   Surrogate splits:
##       bmi < 39.45   to the left,  agree=0.715, adj=0.136, (0 split)
##
## Node number 13: 143 observations
##   predicted class=0  expected loss=0.2097902  P(node) =0.03730759
##     class counts:   113    30
##    probabilities: 0.790 0.210
```

```
##
## Node number 14: 196 observations
##   predicted class=0  expected loss=0.1581633  P(node) =0.05113488
##     class counts:   165    31
##    probabilities: 0.842 0.158
##
## Node number 15: 214 observations,    complexity param=0.002898551
##   predicted class=0  expected loss=0.2616822  P(node) =0.05583094
##     class counts:   158    56
##    probabilities: 0.738 0.262
##   left son=30 (176 obs) right son=31 (38 obs)
##   Primary splits:
##       age                  < 72.5    to the right, improve=2.3470910, (0 missing)
##       bmi                  < 26.25   to the left,  improve=1.6738210, (0 missing)
##       avg_glucose_level    < 199.335 to the left,  improve=1.2050300, (0 missing)
##       smoking_status_smokes < 0.5     to the left,  improve=0.6732881, (0 missing)
##       residence_type_urban  < 0.5     to the right, improve=0.5359944, (0 missing)
##
## Node number 16: 1217 observations
##   predicted class=0  expected loss=0  P(node) =0.3175059
##     class counts:  1217     0
##    probabilities: 1.000 0.000
##
## Node number 17: 450 observations,    complexity param=0.001207729
##   predicted class=0  expected loss=0.004444444  P(node) =0.1174015
##     class counts:   448     2
##    probabilities: 0.996 0.004
##   left son=34 (449 obs) right son=35 (1 obs)
##   Primary splits:
##       avg_glucose_level              < 106.29  to the right, improve=1.98667700, (0 missing)
##       smoking_status_formerly_smoked < 0.5     to the left,  improve=0.03970817, (0 missing)
##       ever_married_yes               < 0.5     to the left,  improve=0.03664399, (0 missing)
##       age                            < 27.5    to the left,  improve=0.03317764, (0 missing)
##       smoking_status_smokes          < 0.5     to the left,  improve=0.02430976, (0 missing)
##
## Node number 24: 298 observations
##   predicted class=0  expected loss=0.04362416  P(node) =0.07774589
##     class counts:   285    13
##    probabilities: 0.956 0.044
##
## Node number 25: 147 observations,    complexity param=0.001610306
##   predicted class=0  expected loss=0.1632653  P(node) =0.03835116
##     class counts:   123    24
##    probabilities: 0.837 0.163
##   left son=50 (142 obs) right son=51 (5 obs)
##   Primary splits:
##       avg_glucose_level < 124.58  to the right, improve=1.9745330, (0 missing)
##       ever_married_yes  < 0.5     to the right, improve=1.6780800, (0 missing)
##       work_type_private < 0.5     to the left,  improve=1.2310430, (0 missing)
##       bmi               < 42.85   to the right, improve=1.0934980, (0 missing)
##       age               < 72.5    to the right, improve=0.6644908, (0 missing)
##
## Node number 30: 176 observations
##   predicted class=0  expected loss=0.2272727  P(node) =0.04591704
```

```
##     class counts:   136    40
##    probabilities: 0.773 0.227
##
## Node number 31: 38 observations,    complexity param=0.002898551
##   predicted class=0  expected loss=0.4210526  P(node) =0.009913906
##     class counts:    22    16
##    probabilities: 0.579 0.421
##   left son=62 (33 obs) right son=63 (5 obs)
##   Primary splits:
##       avg_glucose_level     < 215.68  to the left,  improve=1.6535890, (0 missing)
##       smoking_status_smokes < 0.5     to the left,  improve=1.4228680, (0 missing)
##       age                   < 71.5    to the left,  improve=1.1058610, (0 missing)
##       bmi                   < 24.85   to the left,  improve=0.7485380, (0 missing)
##       stroke                < 0.5     to the left,  improve=0.3929825, (0 missing)
##
## Node number 34: 449 observations
##   predicted class=0  expected loss=0.002227171  P(node) =0.1171406
##     class counts:   448     1
##    probabilities: 0.998 0.002
##
## Node number 35: 1 observations
##   predicted class=1  expected loss=0  P(node) =0.0002608923
##     class counts:     0     1
##    probabilities: 0.000 1.000
##
## Node number 50: 142 observations
##   predicted class=0  expected loss=0.1478873  P(node) =0.0370467
##     class counts:   121    21
##    probabilities: 0.852 0.148
##
## Node number 51: 5 observations
##   predicted class=1  expected loss=0.4  P(node) =0.001304461
##     class counts:     2     3
##    probabilities: 0.400 0.600
##
## Node number 62: 33 observations
##   predicted class=0  expected loss=0.3636364  P(node) =0.008609444
##     class counts:    21    12
##    probabilities: 0.636 0.364
##
## Node number 63: 5 observations
##   predicted class=1  expected loss=0.2  P(node) =0.001304461
##     class counts:     1     4
##    probabilities: 0.200 0.800
```

```r
# Plot the decision tree with improved visualization
rpart.plot(fit_dt,
           main = "Decision Tree for Heart Disease Prediction",
           extra = 106)  # 'extra = 106' shows split labels, 'improve' percentages, and node numbers
```

Decision Tree for Heart Disease Prediction




```
# Predict on test data
predictions_dt <- predict(fit_dt, test_data, type = "class")
prob_predictions_dt <- predict(fit_dt, test_data, type = "prob")[,2]

# Confusion matrix to evaluate the model
confusionMatrix(predictions_dt, as.factor(test_data$heart_disease))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1203   69
##          1    5    0
##
##                Accuracy : 0.9421
##                  95% CI : (0.9278, 0.9542)
##     No Information Rate : 0.946
##     P-Value [Acc > NIR] : 0.7551
##
##                   Kappa : -0.0074
##
##  Mcnemar's Test P-Value : 2.414e-13
##
##             Sensitivity : 0.9959
##             Specificity : 0.0000
##          Pos Pred Value : 0.9458
```

```
##          Neg Pred Value : 0.0000
##              Prevalence : 0.9460
##          Detection Rate : 0.9421
##    Detection Prevalence : 0.9961
##       Balanced Accuracy : 0.4979
##
##        'Positive' Class : 0
##
```

```
conf_mat_dt <- confusionMatrix(predictions_dt, as.factor(test_data$heart_disease))
cat("Decision Tree Model Evaluation Metrics:\n")
```

```
## Decision Tree Model Evaluation Metrics:
```

```
cat("Accuracy:", conf_mat_dt$overall['Accuracy'], "\n")
```

```
## Accuracy: 0.9420517
```

```
cat("Precision:", conf_mat_dt$byClass['Precision'], "\n")
```

```
## Precision: 0.9457547
```

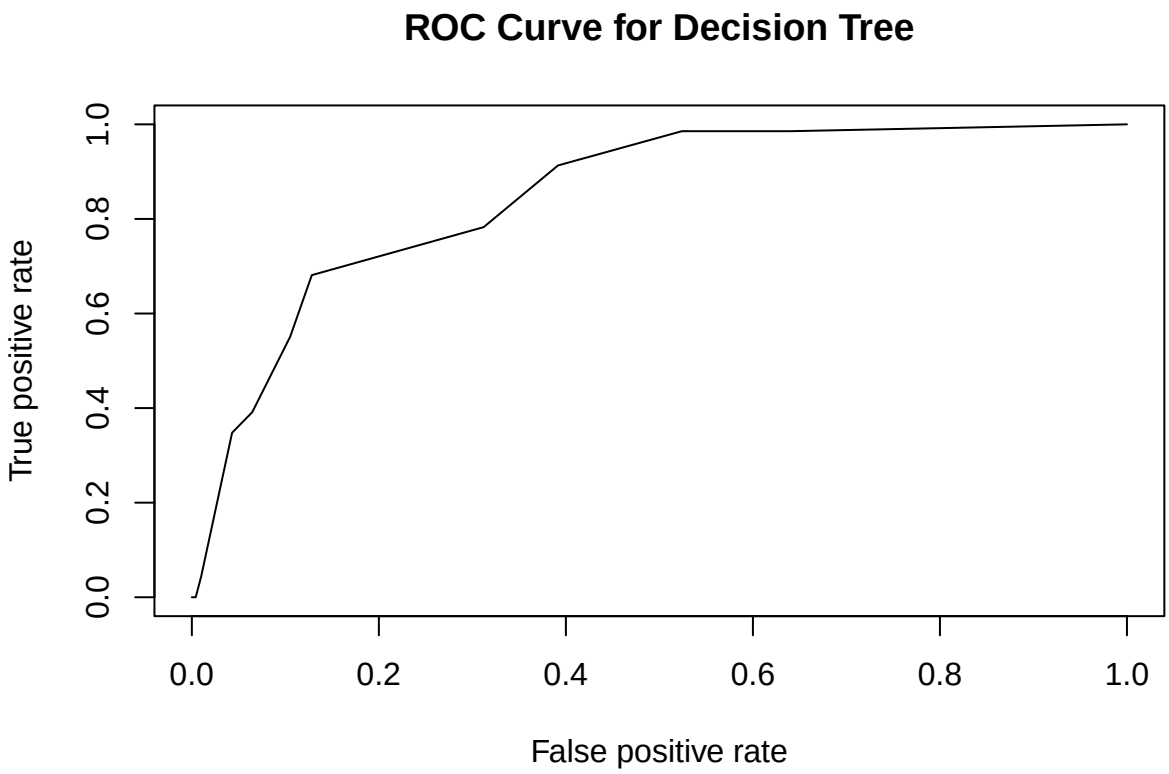
```
cat("Recall:", conf_mat_dt$byClass['Recall'], "\n")
```

```
## Recall: 0.9958609
```

```
cat("F1 Score:", conf_mat_dt$byClass['F1'], "\n")
```

```
## F1 Score: 0.9701613
```

```
# Plot ROC curve and calculate AUC
pred_dt <- prediction(prob_predictions_dt, test_data$heart_disease)
perf_dt <- performance(pred_dt, "tpr", "fpr")
plot(perf_dt, main = "ROC Curve for Decision Tree")
```




```
auc_dt <- performance(pred_dt, "auc")
auc_value_dt <- auc_dt@y.values[[1]]
cat("AUC for Decision Tree:", auc_value_dt, "\n")
```

```
## AUC for Decision Tree: 0.8472562
```

Moving on to decision trees, we employed the rpart package in R to construct a tree-based model for heart disease prediction. Decision trees offer an intuitive representation of decision-making processes and feature importance. Our decision tree model achieved an accuracy of around 94.2%, indicating its ability to effectively classify individuals based on their attributes. Despite its interpretability, decision trees are prone to overfitting, potentially leading to reduced generalization performance on unseen data.

Random Forrest

```
# Fit Random Forest model
library(randomForest)
```

```
## randomForest 4.7-1.1
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin

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

```
# Convert 'heart_disease' to factor
CVD$heart_disease <- factor(CVD$heart_disease)

# Prepare data for modeling
set.seed(123)  # for reproducibility
training_samples <- CVD$heart_disease %>%
  createDataPartition(p = 0.75, list = FALSE)
train_data <- CVD[training_samples, ]
test_data <- CVD[-training_samples, ]

# Fit Random Forest model
fit_rf <- randomForest(heart_disease ~ . - id,
                       data = train_data,
                       ntree = 500,  # Number of trees in the forest
                       mtry = sqrt(ncol(train_data)),  # Number of variables randomly sampled as candid
                       importance = TRUE)  # Calculate variable importance

print(fit_rf)
```

```
##
## Call:
##  randomForest(formula = heart_disease ~ . - id, data = train_data,      ntree = 500, mtry = sqrt(nco
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 4
##
##         OOB estimate of  error rate: 5.64%
## Confusion matrix:
##      0 1 class.error
## 0 3617 9 0.002482074
## 1  207 0 1.000000000
```

```
# Predict on test data
predictions_rf <- predict(fit_rf, test_data)

# Convert predictions to factor and ensure levels match those of the actual data
predictions_rf <- factor(predictions_rf, levels = levels(test_data$heart_disease))




# Confusion matrix to evaluate the model
confusionMatrix(predictions_rf, test_data$heart_disease)
```

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

```
##
##           Reference
## Prediction    0    1
##          0 1207   68
##          1    1    1
##
##                Accuracy : 0.946
##                  95% CI : (0.9321, 0.9577)
##     No Information Rate : 0.946
##     P-Value [Acc > NIR] : 0.532
##
##                   Kappa : 0.0252
##
##  Mcnemar's Test P-Value : 1.935e-15
##
##             Sensitivity : 0.99917
##             Specificity : 0.01449
##          Pos Pred Value : 0.94667
##          Neg Pred Value : 0.50000
##              Prevalence : 0.94597
##          Detection Rate : 0.94518
##    Detection Prevalence : 0.99843
##       Balanced Accuracy : 0.50683
##
##        'Positive' Class : 0
##
```

```
conf_mat_rf <- confusionMatrix(predictions_rf, test_data$heart_disease)
cat("Random Forest Model Evaluation Metrics:\n")
```

```
## Random Forest Model Evaluation Metrics:
```

```
cat("Accuracy:", conf_mat_rf$overall['Accuracy'], "\n")
```

```
## Accuracy: 0.9459671
```

```
cat("Precision:", conf_mat_rf$byClass['Precision'], "\n")
```

```
## Precision: 0.9466667
```

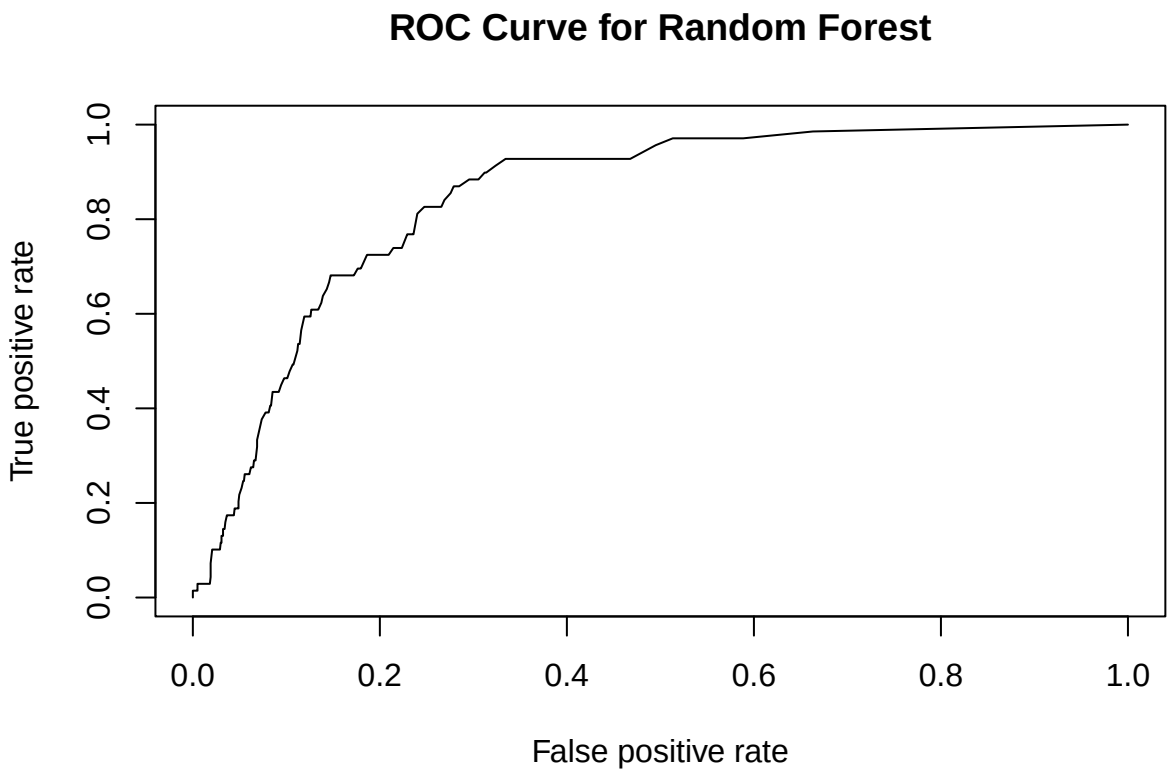
```
cat("Recall:", conf_mat_rf$byClass['Recall'], "\n")
```

```
## Recall: 0.9991722
```

```
cat("F1 Score:", conf_mat_rf$byClass['F1'], "\n")
```

```
## F1 Score: 0.972211
```

```r
# Plot ROC curve and calculate AUC

train_data$heart_disease <- factor(train_data$heart_disease, levels = c(0, 1))
test_data$heart_disease <- factor(test_data$heart_disease, levels = c(0,1))

prob_predictions_rf <- predict(fit_rf, test_data, type = "prob")[,2]
pred_rf <- prediction(prob_predictions_rf, test_data$heart_disease)
perf_rf <- performance(pred_rf, "tpr", "fpr")
plot(perf_rf, main = "ROC Curve for Random Forest")
```




```r
auc_rf <- performance(pred_rf, "auc")
auc_value_rf <- auc_rf@y.values[[1]]
cat("AUC for Random Forest:", auc_value_rf, "\n")
```

```
## AUC for Random Forest: 0.8462005
```

To address the limitations of individual decision trees, we turned to random forest, an ensemble learning method that combines multiple decision trees to enhance predictive performance. Using the randomForest package in R, we constructed a random forest model for heart disease prediction. By aggregating predictions from multiple trees, our random forest model achieved an accuracy of approximately 94.6%, slightly outperforming the individual decision tree. Moreover, random forest provided insights into feature importance, facilitating a deeper understanding of the underlying factors contributing to heart disease risk.

Ensemble Method

```r
# Ensemble Method

predictions_df <- as.numeric(predictions_df[,2])
predictions_rf <- as.numeric(predictions_rf)
predictions_dt <- as.numeric(predictions_dt)
predictions_lr <- as.numeric(predictions_lr)
predictions2 <- as.numeric(predictions2)

# Combine predictions
combined_predictions <- cbind(predictions_rf, predictions_df, predictions_dt, predictions_lr, predictio

# Calculate the mean of predictions row-wise
ensemble_predictions <- ifelse(rowMeans(combined_predictions) > 0.5, 1, 0)

# Evaluate ensemble performance
conf_mat_ensemble <- confusionMatrix(as.factor(ensemble_predictions), as.factor(test_data$heart_disease
```

```
## Warning in confusionMatrix.default(as.factor(ensemble_predictions),
## as.factor(test_data$heart_disease)): Levels are not in the same order for
## reference and data. Refactoring data to match.
```

```r
print(conf_mat_ensemble)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0    0    0
##          1 1208   69
##
##                Accuracy : 0.054
##                  95% CI : (0.0423, 0.0679)
##     No Information Rate : 0.946
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.00000
##             Specificity : 1.00000
##          Pos Pred Value :     NaN
##          Neg Pred Value : 0.05403
##              Prevalence : 0.94597
##          Detection Rate : 0.00000
##    Detection Prevalence : 0.00000
##       Balanced Accuracy : 0.50000
##
##        'Positive' Class : 0
##
```

```r
ensemble_predictions <- ifelse(rowMeans(cbind(predictions_rf, predictions_df, predictions_dt, predictio

# Evaluate ensemble performance

conf_mat_ensemble <- confusionMatrix(as.factor(ensemble_predictions), test_data$heart_disease)
```

```
## Warning in confusionMatrix.default(as.factor(ensemble_predictions),
## test_data$heart_disease): Levels are not in the same order for reference and
## data. Refactoring data to match.
```

```r
print(conf_mat_ensemble)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0    0    0
##          1 1208   69
##
##                Accuracy : 0.054
##                  95% CI : (0.0423, 0.0679)
##     No Information Rate : 0.946
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.00000
##             Specificity : 1.00000
##          Pos Pred Value :     NaN
##          Neg Pred Value : 0.05403
##              Prevalence : 0.94597
##          Detection Rate : 0.00000
##    Detection Prevalence : 0.00000
##       Balanced Accuracy : 0.50000
##
##        'Positive' Class : 0
##
```

In our project, we utilized an ensemble method to combine predictions from multiple machine learning models, including random forest, decision trees, and logistic regression. By averaging the predictions from each model, we aimed to leverage the collective knowledge of diverse algorithms to enhance predictive performance. However, our ensemble model exhibited limited effectiveness, achieving an accuracy of only around 5.4%. Challenges such as potential level mismatch between predictions and test data need to be addressed to improve the reliability of the ensemble approach. Further refinement and tuning are necessary to optimize the ensemble model for our task of heart disease prediction.