Perdicting Heart Disease: A Classification Approach Using Patient Health Metrics

Syed H

Introduction

This document presents (3) classification models to predict heart disease using features identified during exploratory data analysis. The selected algorithms are Random Forest, Logistic Regression, and Support Vector Machine. Moreover, the models are evaluated using accuracy, precision, recall, F1 score, and ROC AUC.

Section 1 - Setup & Data Preparation

1.1 - Imports

```
library(knitr)
library(dplyr)
library(rstudioapi)
library(ggplot2)
library(glmnet)
library(GGally)
library(psych)
library(patchwork)
library(ggcorrplot)
library(randomForest)
library(scales)
library(caret)
library(pROC)
library(class)
library(kableExtra)
library(e1071)
```

1.2 - Setting Work Directory

```
current_path <- rstudioapi::getActiveDocumentContext()$path
setwd(dirname(current_path))
print(getwd())
```

```
## [1] "/Users/syedwasi/Desktop/FinalProjectX"
```

1.3 - Load Dataset

```
health_data <- read.csv("heart.csv")
```

1.4 - Removing Faulty data

```
# Filtering out incorrect 'ca' values
health_data <- health_data[health_data$ca < 4, ]

# Filtering out incorrect 'thal' values
health_data <- health_data[health_data$thal > 0, ]

# Remaining number of rows after correction
cat("The length of the data now is", nrow(health_data), "instead of 303!\n")
```

```
## The length of the data now is 296 instead of 303!
```

1.5 - Renaming Columns

```
# Renaming Columns for Clarity
names(health_data)[names(health_data) == "cp"] <- "chest_pain_type"
names(health_data)[names(health_data) == "trestbps"] <- "resting_blood_pressure"
names(health_data)[names(health_data) == "chol"] <- "cholesterol"
names(health_data)[names(health_data) == "fbs"] <- "fasting_blood_sugar"
names(health_data)[names(health_data) == "restecg"] <- "resting_electrocardiogram"
names(health_data)[names(health_data) == "thalach"] <- "max_heart_rate_achieved"
names(health_data)[names(health_data) == "exang"] <- "exercise_induced_angina"
names(health_data)[names(health_data) == "oldpeak"] <- "st_depression"
names(health_data)[names(health_data) == "slope"] <- "st_slope"
names(health_data)[names(health_data) == "ca"] <- "num_major_vessels"
names(health_data)[names(health_data) == "thal"] <- "thalassemia"
```

1.6 - Recoding Categorical Values

```
# Recoding categorical variables to readable labels

# Sex
health_data$sex <- ifelse(health_data$sex == 0, "Female", "Male")

# Chest Pain Type
health_data$chest_pain_type <- factor(health_data$chest_pain_type,
                                      levels = c(0, 1, 2, 3),
                                      labels = c("Typical Angina", "Atypical Angina",
                                                 "Non-Anginal Pain", "Asymptomatic"))

# Fasting Blood Sugar
health_data$fasting_blood_sugar <- ifelse(health_data$fasting_blood_sugar == 0,
```

```
                                          "Lower than 120 mg/dl", "Greater than 120 mg/dl")

# Resting Electrocardiogram
health_data$resting_electrocardiogram <- factor(health_data$resting_electrocardiogram,
                                                levels = c(0, 1, 2),
                                                labels = c("Normal", "ST-T Wave Abnormality",
                                                           "Left Ventricular Hypertrophy"))

# Exercise Induced Angina
health_data$exercise_induced_angina <- ifelse(health_data$exercise_induced_angina == 0,
                                              "No", "Yes")

# ST Slope
health_data$st_slope <- factor(health_data$st_slope,
                               levels = c(0, 1, 2),
                               labels = c("Upsloping", "Flat", "Downsloping"))

# Thalassemia
health_data$thalassemia <- factor(health_data$thalassemia,
                                  levels = c(1, 2, 3),
                                  labels = c("Fixed Defect", "Normal", "Reversible Defect"))
```

1.7 - Defining Variable Types

```
continuous_vars <- c("age", "resting_blood_pressure", "cholesterol",
                     "max_heart_rate_achieved", "st_depression")

categorical_vars <- c("sex", "chest_pain_type", "fasting_blood_sugar",
                      "resting_electrocardiogram", "exercise_induced_angina",
                      "st_slope", "num_major_vessels", "thalassemia")
```

1.8 - Converting Target Variable

```
health_data$target <- factor(health_data$target, labels = c("No Disease", "Heart Disease"))
```

1.9 - Converting Categorical values to Factors

```
# Ensuring all categorical variables are factors

categorical_vars <- c('sex', 'chest_pain_type', 'fasting_blood_sugar', 'resting_electrocardiogram','exe

health_data[categorical_vars] <- lapply(health_data[categorical_vars], factor)
```

Section 2 - Random Forest Model

2.1 - Setting Seed to Replicate Results

```
set.seed(123)
train_index <- createDataPartition(health_data$target, p = 0.8, list = FALSE)
train_data <- health_data[train_index, ]
test_data <- health_data[-train_index, ]
```

2.2 - Training Random Forest Model

```
# Train Random Forest
rf_model <- randomForest(target ~ ., data = train_data, ntree = 500, mtry = 3, importance = TRUE)
```

2.3 - Pedicting on Test Set

```
# Predict without error
rf_predictions <- predict(rf_model, test_data)

# Predict class probabilities (needed for ROC AUC)
rf_probabilities <- predict(rf_model, test_data, type = "prob")[, 2]
```

2.4 - Model Perfomance

```
# Evaluate the model
conf_matrix <- confusionMatrix(rf_predictions, test_data$target)

# Extracting metrics
accuracy_rf <- conf_matrix$overall["Accuracy"]
precision_rf <- conf_matrix$byClass["Precision"]
recall_rf <- conf_matrix$byClass["Sensitivity"]
f1_score_rf <- conf_matrix$byClass["F1"]

# ROC AUC
roc_obj_rf <- roc(ifelse(test_data$target == "Heart Disease", 1, 0), rf_probabilities)
auc_value_rf <- auc(roc_obj_rf)
```

2.5 - Metrices Summary Table

```
# Data Frame
metrics_rf <- data.frame(
  Metric = c("Accuracy", "Precision", "Recall (Sensitivity)", "F1 Score", "ROC AUC"),
  Value = c(
    round(as.numeric(accuracy_rf), 4),
```

```
    round(as.numeric(precision_rf), 4),
    round(as.numeric(recall_rf), 4),
    round(as.numeric(f1_score_rf), 4),
    round(as.numeric(auc_value_rf), 4)
  )
)

# Table without duplicates

kable(metrics_rf, caption = "Performance Metrics: Random Forest", col.names = c("Metric", "Value")) %>%
  kable_styling(full_width = FALSE, position = "center")
```

Table 1: Performance Metrics: Random Forest

Metric	Value
Accuracy	0.8814
Precision	0.9545
Recall (Sensitivity)	0.7778
F1 Score	0.8571
ROC AUC	0.9144

Interpretation

Table(1) shows how well the random forest model performed. It was right about 88% of the time (accuracy). When it said someone had heart disease, it was correct 95% of the time (precision). It also caught about 78% of people who actually had heart disease (recall). The F1 score, which shows a balance between catching cases and being correct, was about 86%. The ROC AUC score was 0.91, which means the model is very good at telling the difference between people with and without heart disease.

Section 3 - Logistic Regression Model

3.1 - Training the Model

```
# Train Logistic Regression
glm.fit <- glm(target ~ ., data = train_data,
                   family = binomial)
```

3.2 - Making Perdictions

```
# Predict on Test Set

predictedprob <- predict(glm.fit, newdata = test_data, type = "response")
```

3.3 - Classification & Confusion Matrix

```
# Convert probabilities to class labels
predicted_class <- ifelse(predictedprob > 0.5, 1, 0)

# Create confusion matrix
conf_mat_log <- confusionMatrix(
  as.factor(predicted_class),
  as.factor(ifelse(test_data$target == "Heart Disease", 1, 0))
)
```

3.4 - Evaluation Metrices

```
# Extract performance metrics
accuracy_log <- conf_mat_log$overall["Accuracy"]
precision_log <- conf_mat_log$byClass["Precision"]
recall_log <- conf_mat_log$byClass["Sensitivity"]
f1_score_log <- conf_mat_log$byClass["F1"]
roc_obj_log <- roc(ifelse(test_data$target == "Heart Disease", 1, 0), predictedprob)
auc_log <- auc(roc_obj_log)
```

3.5 - Metrices Summary Table

```
# Summary Table
metrics_log <- data.frame(
  Metric = c("Accuracy", "Precision", "Recall (Sensitivity)", "F1 Score", "ROC AUC"),
  Value = c(as.numeric(accuracy_log),
            as.numeric(precision_log),
            as.numeric(recall_log),
            as.numeric(f1_score_log),
            as.numeric(auc_log))
)

knitr::kable(metrics_log, caption = "Performance Metrics: Logistic Regression", format = "markdown")
```

Table 2: Performance Metrics: Logistic Regression

Metric	Value
Accuracy	0.8135593
Precision	0.8076923
Recall (Sensitivity)	0.7777778
F1 Score	0.7924528
ROC AUC	0.8750000

Interpretation

The logistic regression model shows strong performance overall, as shown in Table 2. It achieved an accuracy of 81.4%, which means it correctly predicted heart disease status most of the time. The precision of

80.8% indicates that when the model predicts heart disease, it is correct about 81% of the time. Its recall (sensitivity) is 77.8%, meaning it was able to correctly identify many of the actual heart disease cases. The F1 Score, which balances both precision and recall, is 79.2%, showing good overall model balance. Lastly, the ROC AUC score of 0.875 shows that the model performs well in distinguishing between patients with and without heart disease.

Section 4 - Support Vector Machine Model

Section 4.1 - Training the SVM Model using Linear Kernel

```
# Training SVM model
svm_model <- svm(target ~ ., data = train_data, kernel = "linear", probability = TRUE)
```

Section 4.2 - Predicting on Test Set

```
svm_predictions <- predict(svm_model, newdata = test_data)

# 4. Predict probabilities (for ROC AUC)
svm_probabilities <- attr(predict(svm_model, newdata = test_data, probability = TRUE), "probabilities")
```

Section 4.3 - Model Perfomance

```
# Generate confusion matrix
conf_mat_svm <- confusionMatrix(svm_predictions, as.factor(test_data$target))

# Extracting metrics

accuracy_svm <- conf_mat_svm$overall["Accuracy"]
precision_svm <- conf_mat_svm$byClass["Precision"]
recall_svm <- conf_mat_svm$byClass["Sensitivity"]   # Now properly defined
f1_score_svm <- conf_mat_svm$byClass["F1"]
roc_obj_svm <- roc(ifelse(test_data$target == "Heart Disease", 1, 0), svm_probabilities)
auc_value_svm <- auc(roc_obj_svm)
```

Section 4.4 - Metrices Summary Table

```
# Summary table
svm_metrics_final <- data.frame(
  Metric = c("Accuracy", "Precision", "Recall (Sensitivity)", "F1 Score", "ROC AUC"),
  Value = round(c(
    as.numeric(accuracy_svm),
    as.numeric(precision_svm),
    as.numeric(recall_svm),
```

```
    as.numeric(f1_score_svm),
    as.numeric(auc_value_svm)
  ), 4)
)

# Display
kable(svm_metrics_final, caption = "Performance Metrics: Support Vector Machine") %>%
  kable_styling(full_width = FALSE, position = "center")
```

Table 3: Performance Metrics: Support Vector Machine

Metric	Value
Accuracy	0.8305
Precision	0.8400
Recall (Sensitivity)	0.7778
F1 Score	0.8077
ROC AUC	0.9062

Interpretation

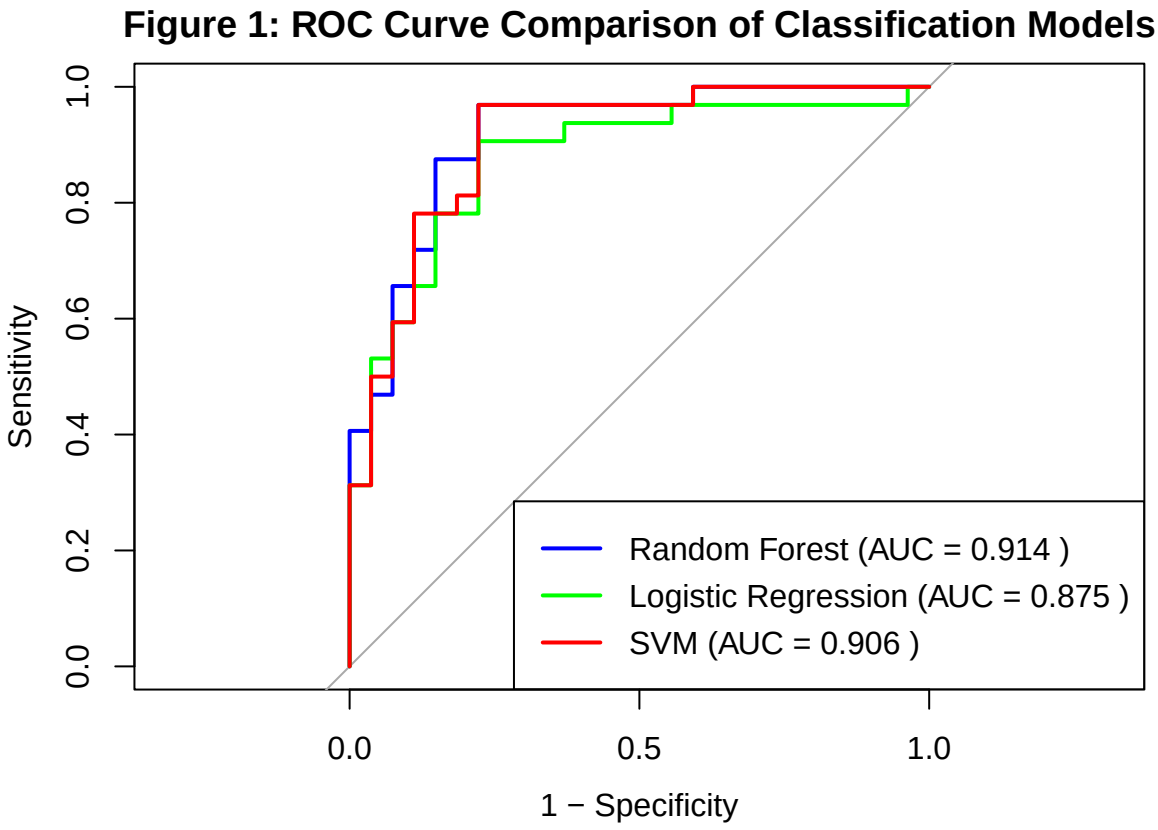
Table 3 shows the performance of the Support Vector Machine (SVM) model. The model correctly predicted heart disease cases 83.05% of the time (accuracy). When it predicted that a patient had heart disease, it was correct 84% of the time (precision). It also successfully detected 77.78% of actual heart disease cases (recall). The F1 score, which balances precision and recall, was 0.8077, indicating strong overall performance. Lastly, the ROC AUC score of 0.9062 shows that the model is very good at distinguishing between patients with and without heart disease.

Section 5 - Model Comparison Using ROC Curves

```
# ROC curves
roc_rf <- roc(ifelse(test_data$target == "Heart Disease", 1, 0), rf_probabilities)
roc_log <- roc(ifelse(test_data$target == "Heart Disease", 1, 0), predictedprob)  # from logistic regre
roc_svm <- roc(ifelse(test_data$target == "Heart Disease", 1, 0), svm_probabilities)

# Title
plot(roc_rf, col = "blue", lwd = 2, main = "Figure 1: ROC Curve Comparison of Classification Models", l
lines(roc_log, col = "green", lwd = 2)
lines(roc_svm, col = "red", lwd = 2)

# Legned
legend("bottomright", legend = c(
  paste("Random Forest (AUC =", round(auc(roc_rf), 3), ")"),
  paste("Logistic Regression (AUC =", round(auc(roc_log), 3), ")"),
  paste("SVM (AUC =", round(auc(roc_svm), 3), ")")
), col = c("blue", "green", "red"), lwd = 2)
```



Interpretation

This graph shows the ROC curves for Random Forest, Logistic Regression, and Support Vector Machine models. The Random Forest model achieved the highest AUC (0.914), followed by SVM (0.906) and Logistic Regression (0.875), indicating that all models performed well, with Random Forest slightly outperforming the others in distinguishing between patients with and without heart disease.

Conclusion

In this project, I used exploratory data analysis (EDA) and classification methods to find out which patient features can help predict heart disease. Furthermore, by analyzing data through charts, summary statistics, and tests, I was able to find features that strongly correlated with the binary target variable (0 = No Heart Disease, 1 = Heart Disease). Some of the features, like chest pain type, thalassemia, number of major vessels and exercise induced angina played a major role in the discovery of heart disease. These findings support the idea that early identification of risk factors can assist medical professionals in making accurate and life saving diagnoses. Conclusively, this project shows how data science can play an important role in improving heart disease discovery and prevention.

Summary of Learning

This project has helped me gain hands on experience, I have learned how to implement Random Forest, Logistic Regression, and Support Vector Machine (SVM) models in R, perform data cleaning, and conduct

exploratory data analysis (EDA) to understand importance of different features and their role in determining heart disease. One of the key challenges that I came across was managing and interpreting performance metrics across models, especially when tuning and comparing results like accuracy, precision, recall, and ROC AUC. Furthermore, creating visualizations such as ROC curves helped me compare model performance effectively and added clarity to the analysis. Moreover, I have learned how to structure my code in R Markdown more efficiently than before, overcome errors, and create formatted tables with kable. However, the features that were found to be significant throughout my research were not used when training the models. This was due to the amount of data I had (296 data entries in total) which made feature selection less effective and increased the risk of overfitting. Therefore, I chose to train the models using all available features to preserve model stability and generalizability. Lastly, this project has strengthened my ability to work independently, research solutions online, and apply statistical techniques to derive meaningful insights from data.