Task 2: Predicting Heart Disease using Decision Trees and Support Vector Machines

```
library(readxl)
library(rpart)
library(rpart.plot)
library(caret)
library(tidyverse)
library(e1071)
library(caTools)

file <- "/Users/ny/COSC 3337/Predicting_Heart_Disease/heart-disease2.xlsx"
data <- read_excel(file)
```

```
set.seed(123)

split <- sample.split(data$target, SplitRatio = 0.8)
train_data <- subset(data, split == TRUE)
test_data <- subset(data, split == FALSE)
```

```
tree_depth_3 <- rpart(target ~ ., data = train_data, method = "class", control = rpart.contr
predictions_depth_3 <- predict(tree_depth_3, test_data, type = "class")
tree_depth_7 <- rpart(target ~ ., data = train_data, method = "class", control = rpart.contr
predictions_depth_7 <- predict(tree_depth_7, test_data, type = "class")
tree_depth_11 <- rpart(target ~ ., data = train_data, method = "class", control = rpart.cont
predictions_depth_11 <- predict(tree_depth_11, test_data, type = "class")
tree_depth_15 <- rpart(target ~ ., data = train_data, method = "class", control = rpart.cont
predictions_depth_15 <- predict(tree_depth_15, test_data, type = "class")
```

```r
train_control <- trainControl(method = "cv", number = 5)

train_data$target <- as.factor(train_data$target)
test_data$target <- as.factor(test_data$target)

compute_metrics <- function(max_depth) {
  model <- train(target ~ .,
                 data = train_data,
                 method = "rpart",
                 trControl = train_control,
                 tuneGrid = data.frame(cp = 0),
                 control = rpart.control(maxdepth = max_depth))

  predictions <- predict(model, newdata = test_data)

  predictions <- factor(predictions, levels = levels(test_data$target))
  test_target <- factor(test_data$target, levels = levels(test_data$target))


  confusion_matrix <- confusionMatrix(predictions, test_target)

  accuracy <- confusion_matrix$overall['Accuracy']
  precision <- confusion_matrix$byClass['Pos Pred Value']
  recall <- confusion_matrix$byClass['Sensitivity']

  return(c(accuracy, precision, recall))
}

depths <- c(3, 7, 11, 15)

results <- data.frame(Max_Depth = depths, Accuracy = numeric(4), Precision = numeric(4), Reca

for (i in 1:length(depths)) {
  metrics <- compute_metrics(depths[i])
  results[i, 2:4] <- metrics
}

results
```

```
  Max_Depth  Accuracy Precision    Recall
1         3 0.7049180 0.7083333 0.6071429
2         7 0.7540984 0.8421053 0.5714286
```

```
3         11 0.7540984 0.8421053 0.5714286
4         15 0.7540984 0.8421053 0.5714286
```

In this particular model, Max Depth 3 seems to underfit the data, leading to lower accuracy and precision, but slightly higher recall.

Max Depth 7 is balanced as it has a better accuracy and precision but a slightly lower recall. It doesn't change from Max Depth 11 or 15 which goes to show that we're not really overfitting the model with these depths.

There are no huge significant differences in the accuracy, precision and recall scores.

```
data$target <- as.factor(data$target)
set.seed(123)
train_index <- sample(1:nrow(data), 0.8 * nrow(data))
train_data <- data[train_index, ]
test_data <- data[-train_index, ]
```

```
svm_linear <- svm(target ~ ., data = train_data, kernel = "linear")

predictions_linear <- predict(svm_linear, test_data)

confusion_matrix_linear <- table(predictions_linear, test_data$target)
accuracy_linear <- sum(diag(confusion_matrix_linear)) / sum(confusion_matrix_linear)
```

```
svm_poly <- svm(target ~ ., data = train_data, kernel = "polynomial")

predictions_poly <- predict(svm_poly, test_data)

confusion_matrix_poly <- table(predictions_poly, test_data$target)
accuracy_poly <- sum(diag(confusion_matrix_poly)) / sum(confusion_matrix_poly)
```

```
svm_sigmoid <- svm(target ~ ., data = train_data, kernel = "sigmoid")

predictions_sigmoid <- predict(svm_sigmoid, test_data)

confusion_matrix_sigmoid <- table(predictions_sigmoid, test_data$target)
accuracy_sigmoid <- sum(diag(confusion_matrix_sigmoid)) / sum(confusion_matrix_sigmoid)
```

```
svm_sigmoid_custom <- svm(target ~ ., data = train_data, kernel = "sigmoid", gamma = 0.5)

predictions_sigmoid_custom <- predict(svm_sigmoid_custom, test_data)
```

```r
confusion_matrix_sigmoid_custom <- table(predictions_sigmoid_custom, test_data$target)
accuracy_sigmoid_custom <- sum(diag(confusion_matrix_sigmoid_custom)) / sum(confusion_matrix_
```

```r
cat("Accuracy (Linear Kernel):", accuracy_linear, "\n")
```

```
Accuracy (Linear Kernel): 0.852459
```

```r
cat("Accuracy (Polynomial Kernel):", accuracy_poly, "\n")
```

```
Accuracy (Polynomial Kernel): 0.7704918
```

```r
cat("Accuracy (Sigmoid Kernel):", accuracy_sigmoid, "\n")
```

```
Accuracy (Sigmoid Kernel): 0.8360656
```

```r
cat("Accuracy (Sigmoid Kernel with custom gamma):", accuracy_sigmoid_custom, "\n")
```

```
Accuracy (Sigmoid Kernel with custom gamma): 0.7213115
```

```r
data$target <- factor(data$target, levels = c("0", "1"))
svm_experiment <- function(kernel_type, gamma_value = NULL) {

  if (kernel_type == "linear") {
    method <- "svmLinear"
  } else if (kernel_type == "polynomial") {
    method <- "svmPoly"
  } else if (kernel_type == "sigmoid") {
    method <- "svmRadial"  # Using RBF kernel with default gamma for sigmoid-like behavior
  } else if (kernel_type == "sigmoid_custom") {
    method <- "svmRadial"  # Use the radial basis function with custom gamma for sigmoid
  }

  # Train SVM based on kernel type and custom gamma if provided
  if (!is.null(gamma_value)) {
    model <- train(target ~ .,
                   data = data,
                   method = method,
                   trControl = train_control,
```

```r
                    tuneGrid = data.frame(C = 1, sigma = gamma_value))  # Custom gamma (sigma
  } else {
    model <- train(target ~ .,
                    data = data,
                    method = method,
                    trControl = train_control)
  }

  # Predictions and confusion matrix
  predictions <- predict(model, data)
  confusion_matrix <- confusionMatrix(predictions, data$target)

  # Extract metrics
  accuracy <- confusion_matrix$overall["Accuracy"]
  precision <- confusion_matrix$byClass["Pos Pred Value"]
  recall <- confusion_matrix$byClass["Sensitivity"]

  return(c(accuracy, precision, recall))
}
```

```r
metrics_linear <- svm_experiment("linear")
metrics_poly <- svm_experiment("polynomial")
metrics_sigmoid <- svm_experiment("sigmoid")
metrics_sigmoid_custom <- svm_experiment("sigmoid", gamma_value = 0.5)
```

```r
results <- data.frame(
  Kernel_Function = c("linear", "polynomial", "sigmoid", "sigmoid with different   value"),
  Accuracy = c(metrics_linear[1], metrics_poly[1], metrics_sigmoid[1], metrics_sigmoid_custo
  Precision = c(metrics_linear[2], metrics_poly[2], metrics_sigmoid[2], metrics_sigmoid_cust
  Recall = c(metrics_linear[3], metrics_poly[3], metrics_sigmoid[3], metrics_sigmoid_custom[
)

print(results)
```

```
                 Kernel_Function  Accuracy Precision    Recall
1                         linear 0.8580858 0.8991597 0.7753623
2                     polynomial 0.8514851 0.9115044 0.7463768
3                        sigmoid 0.8844884 0.9186992 0.8188406
4 sigmoid with different   value 1.0000000 1.0000000 1.0000000
```

The linear kernel performs well with high accuracy but lower recall. The polynomial kernel is the worst performing kernel. The sigmoid kernel outperforms the other two in terms of accuracy, precision and recall. This is useful for capturing complex relationships compared to the other two. However the last kernel is perfect, which is an odd sign.

The linear kernel's **accuracy** and **precision** are relatively high, but the **recall** is lower, indicating that the linear kernel is conservative in predicting positive cases.

The polynomial kernel's **precision** is slightly higher than the linear kernel, meaning the polynomial kernel makes even fewer false positive errors, but **recall** is lower.

The **sigmoid kernel** shows the highest accuracy among the non-overfitting models, and it strikes a good balance between **precision (91.53%)** and **recall (78.26%)**.

The custom sigmoid kernel with perfect accuracy, precision, and recall likely suffers from overfitting.

Model Performance/Comparison

SVM (Sigmoid Kernel) clearly performs better than the Decision Tree in terms of accuracy, precision, and recall. The accuracy, precision, and recall are all better on the SVM. The Decision Tree performs reasonably well, but its performance plateaus at higher depths