DSCI5340_HW4_Group10

```
#Adjusting Seed for reproducibility
set.seed(42)
```

Q1. Import the heart disease data. Make any necessary pre-processing before moving on

the next step

```
heart_data <- read_csv("heart_disease.csv")
```

```
## Rows: 300 Columns: 14
## ── Column specification ────────────────────────────────────────────────────────
## Delimiter: ","
## dbl (14): age, sex, cp, trestbps, chol, fbs, rest_ecg, thalach, exang, oldpe...
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
#dimensions and structure of the data
dim(heart_data)
```

```
## [1] 300  14
```

```
str(heart_data)
```

```
## spc_tbl_ [300 × 14] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ age         : num [1:300] 63 67 67 37 41 56 62 57 63 53 ...
##  $ sex         : num [1:300] 1 1 1 1 0 1 0 0 1 1 ...
##  $ cp          : num [1:300] 1 4 4 3 2 2 4 4 4 4 ...
##  $ trestbps    : num [1:300] 145 160 120 130 130 120 140 120 130 140 ...
##  $ chol        : num [1:300] 233 286 229 250 204 236 268 354 254 203 ...
##  $ fbs         : num [1:300] 1 0 0 0 0 0 0 0 1 ...
##  $ rest_ecg    : num [1:300] 2 2 2 0 2 0 2 0 2 2 ...
##  $ thalach     : num [1:300] 150 108 129 187 172 178 160 163 147 155 ...
##  $ exang       : num [1:300] 0 1 1 0 0 0 0 1 0 1 ...
##  $ oldpeak     : num [1:300] 2.3 1.5 2.6 3.5 1.4 0.8 3.6 0.6 1.4 3.1 ...
##  $ slope       : num [1:300] 3 2 2 3 1 1 3 1 2 3 ...
##  $ ca          : num [1:300] 0 3 2 0 0 0 2 0 1 0 ...
##  $ thal        : num [1:300] 6 3 7 3 3 3 3 3 7 7 ...
##  $ heartdisease: num [1:300] 0 1 1 0 0 0 1 0 1 1 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   age = col_double(),
##   ..   sex = col_double(),
##   ..   cp = col_double(),
##   ..   trestbps = col_double(),
##   ..   chol = col_double(),
##   ..   fbs = col_double(),
##   ..   rest_ecg = col_double(),
##   ..   thalach = col_double(),
##   ..   exang = col_double(),
##   ..   oldpeak = col_double(),
##   ..   slope = col_double(),
##   ..   ca = col_double(),
##   ..   thal = col_double(),
##   ..   heartdisease = col_double()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

```
heart_data$sex <- as.factor(heart_data$sex)
heart_data$cp <- as.factor(heart_data$cp)
heart_data$fbs <- as.factor(heart_data$fbs)
heart_data$rest_ecg <- as.factor(heart_data$rest_ecg)
heart_data$exang <- as.factor(heart_data$exang)
heart_data$thal <- as.factor(heart_data$thal)
heart_data$heartdisease <- as.factor(heart_data$heartdisease)
```

Q2. Partition the data into two parts: training (80%)

and test (20%)

```
set.seed(42)
#library caret is already loaded
partition_index <- createDataPartition(heart_data$heartdisease, p=0.80, list=FALSE)
training_set <- heart_data[partition_index,]
testing_set <- heart_data[-partition_index,]
#dimensions of testing and training datasets
dim(testing_set)
```

```
## [1] 59 14
```

```
dim(training_set)
```

```
## [1] 241  14
```

Q3-Q5 using only one code chunk

```
# Train SVM model
svm_model <- svm(heartdisease ~ ., data = training_set,kernel="linear")
svm_model
```

```
##
## Call:
## svm(formula = heartdisease ~ ., data = training_set, kernel = "linear")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  1
##
## Number of Support Vectors:  82
```

```
# Setup for cross-validation
control <- trainControl(method = "repeatedcv", number = 10, repeats = 3)

# Duplicate datasets for preprocessing
training_preprocessed <- training_set
testing_preprocessed <- testing_set
normalized_data <- heart_data

# Preprocess the data
set.seed(42)
normalization_values <- preProcess(training_set[, c(1,4,5,8,10,11,12)], method=c("center", "scal
e"))
training_preprocessed[, c(1,4,5,8,10,11,12)] <- predict(normalization_values, training_set[, c
(1,4,5,8,10,11,12)])
testing_preprocessed[, c(1,4,5,8,10,11,12)] <- predict(normalization_values, testing_set[, c(1,
4,5,8,10,11,12)])
normalized_data[, c(1,4,5,8,10,11,12)] <- predict(normalization_values, heart_data[, c(1,4,5,8,1
0,11,12)])
```

Q6. Now using the model above, generate the confusion matrix for the test data. What is

the sensitivity from this model?

```
# Model predictions
predictions <- predict(svm_model, newdata = testing_preprocessed)
# Confusion matrix
conf_matrix <- table(Predicted = predictions, Actual = testing_set$heartdisease)
confusionMatrix(conf_matrix)
```

```
## Confusion Matrix and Statistics
##
##          Actual
## Predicted  0  1
##         0 13  1
##         1 19 26
##
##                Accuracy : 0.661
##                  95% CI : (0.526, 0.779)
##     No Information Rate : 0.542
##     P-Value [Acc > NIR] : 0.043593
##
##                   Kappa : 0.351
##
##  Mcnemar's Test P-Value : 0.000144
##
##             Sensitivity : 0.406
##             Specificity : 0.963
##          Pos Pred Value : 0.929
##          Neg Pred Value : 0.578
##              Prevalence : 0.542
##          Detection Rate : 0.220
##    Detection Prevalence : 0.237
##       Balanced Accuracy : 0.685
##
##        'Positive' Class : 0
##
```

```
# Calculate sensitivity
sensitivity_measure <- conf_matrix["1", "1"] / sum(conf_matrix["1", ])
sensitivity_measure
```

```
## [1] 0.578
```

Sesitivity of the above model is 0.406 specificity of above model is 0.963 Accuracy is 0.661

Q7. Run a second SVM model using the grid search hyperparameter optimization

method for C. For this run, choose all values between 0 and 2.5 (both included) with an increment of 0.1

```
set.seed(42)
# Prepare grid for hyperparameter tuning
parameter_grid <- expand.grid(C = seq(0, 2.5, by = 0.1))
# SVM model with grid search
optimized_svm <- train(
  heartdisease ~ .,
  data = training_preprocessed,
  method = "svmLinear",
  trControl = control,
  tuneGrid = parameter_grid
)
# Display optimized model
optimized_svm
```

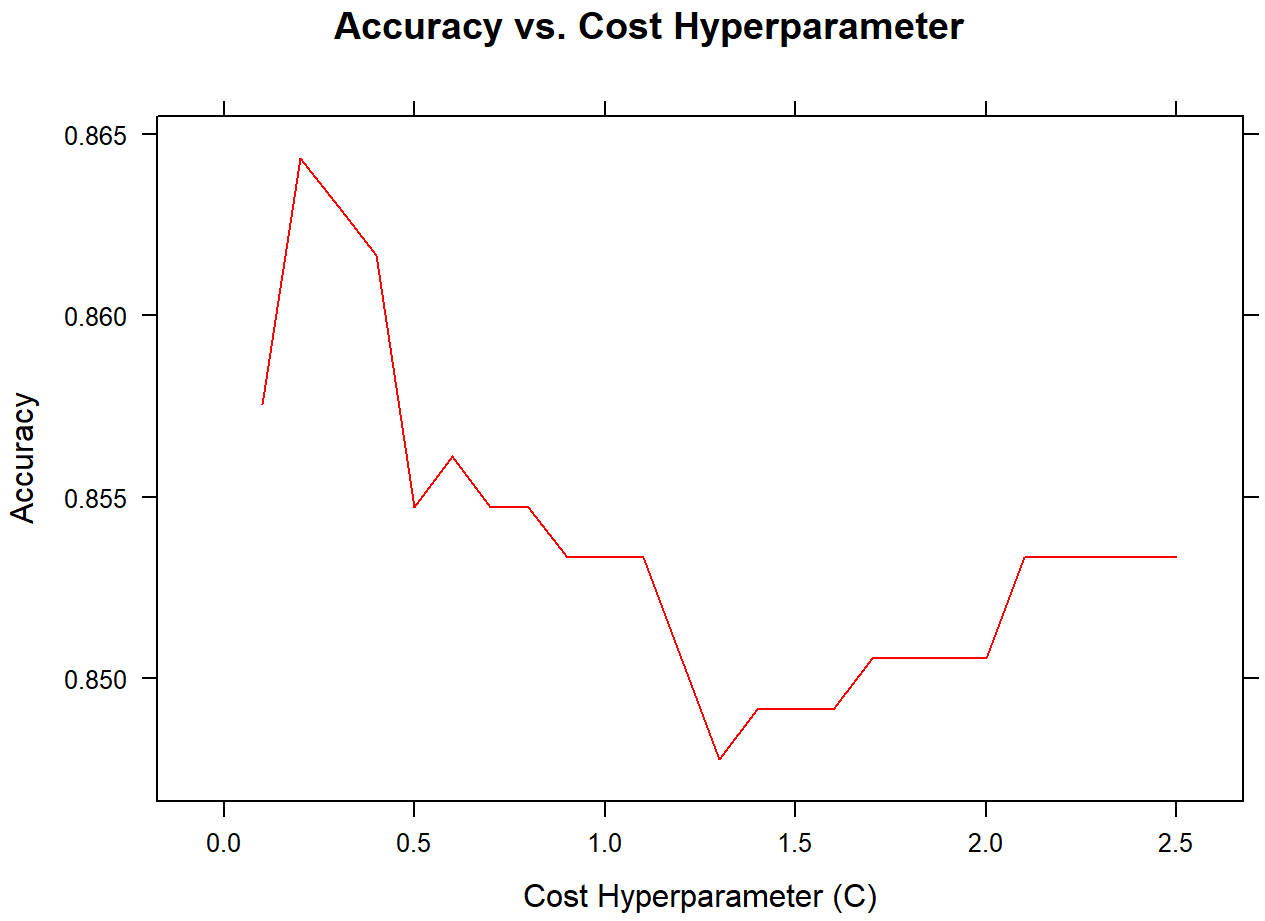
```
## Support Vector Machines with Linear Kernel
##
## 241 samples
##  13 predictor
##   2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 217, 217, 217, 217, 217, 217, ...
## Resampling results across tuning parameters:
##
##   C    Accuracy  Kappa
##   0.0    NaN       NaN
##   0.1  0.858     0.711
##   0.2  0.864     0.725
##   0.3  0.863     0.722
##   0.4  0.862     0.719
##   0.5  0.855     0.705
##   0.6  0.856     0.708
##   0.7  0.855     0.705
##   0.8  0.855     0.705
##   0.9  0.853     0.702
##   1.0  0.853     0.702
##   1.1  0.853     0.702
##   1.2  0.851     0.697
##   1.3  0.848     0.691
##   1.4  0.849     0.694
##   1.5  0.849     0.694
##   1.6  0.849     0.694
##   1.7  0.851     0.697
##   1.8  0.851     0.697
##   1.9  0.851     0.697
##   2.0  0.851     0.697
##   2.1  0.853     0.703
##   2.2  0.853     0.703
##   2.3  0.853     0.703
##   2.4  0.853     0.703
##   2.5  0.853     0.703
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was C = 0.2.
```

The Accuracy is maximum at c=0.2,so we the final value used for this model is at c =0.2

Q8. Generate a plot to examine the relationship between accuracy and the cost

hyperparameter in the second SVM model

```
xyplot(Accuracy ~ C, data = optimized_svm$results, type = "l", main = "Accuracy vs. Cost Hyperpa
rameter", xlab = "Cost Hyperparameter (C)", ylab = "Accuracy",col= "red")
```



Q9. Generate a confusion matrix using the latest model. What is the sensitivity from this model?

```
# Predictions using optimized model
predictions_optimized <- predict(optimized_svm, newdata = testing_preprocessed)
# Confusion matrix for optimized model
conf_matrix_optimized <- confusionMatrix(predictions_optimized, testing_set$heartdisease)
conf_matrix_optimized
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 27  9
##          1  5 18
##
##                Accuracy : 0.763
##                  95% CI : (0.634, 0.864)
##     No Information Rate : 0.542
##     P-Value [Acc > NIR] : 0.000396
##
##                   Kappa : 0.516
##
##  Mcnemar's Test P-Value : 0.422678
##
##             Sensitivity : 0.844
##             Specificity : 0.667
##          Pos Pred Value : 0.750
##          Neg Pred Value : 0.783
##              Prevalence : 0.542
##          Detection Rate : 0.458
##    Detection Prevalence : 0.610
##       Balanced Accuracy : 0.755
##
##        'Positive' Class : 0
##
```

```
# Sensitivity from the optimized model
sensitivity_optimized <- conf_matrix_optimized$sensitivity
```

Sensitivity of the model : 0.844 Specificity of the model : 0.667 Accuracy of the model : 0.763

Q10. Has the performance of the model improved with grid search? Explain using

numbers from the confusion matrices from both models.

```
conf_matrix_initial <- confusionMatrix(predictions, testing_set$heartdisease)
conf_matrix_optimized <- confusionMatrix(predictions_optimized, testing_set$heartdisease)
# Display both confusion matrices
print(conf_matrix_initial)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 13  1
##          1 19 26
##
##                Accuracy : 0.661
##                  95% CI : (0.526, 0.779)
##     No Information Rate : 0.542
##     P-Value [Acc > NIR] : 0.043593
##
##                   Kappa : 0.351
##
##  Mcnemar's Test P-Value : 0.000144
##
##             Sensitivity : 0.406
##             Specificity : 0.963
##          Pos Pred Value : 0.929
##          Neg Pred Value : 0.578
##              Prevalence : 0.542
##          Detection Rate : 0.220
##    Detection Prevalence : 0.237
##       Balanced Accuracy : 0.685
##
##        'Positive' Class : 0
##
```

```
print(conf_matrix_optimized)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 27  9
##          1  5 18
##
##                Accuracy : 0.763
##                  95% CI : (0.634, 0.864)
##     No Information Rate : 0.542
##     P-Value [Acc > NIR] : 0.000396
##
##                   Kappa : 0.516
##
##  Mcnemar's Test P-Value : 0.422678
##
##             Sensitivity : 0.844
##             Specificity : 0.667
##          Pos Pred Value : 0.750
##          Neg Pred Value : 0.783
##              Prevalence : 0.542
##          Detection Rate : 0.458
##    Detection Prevalence : 0.610
##       Balanced Accuracy : 0.755
##
##        'Positive' Class : 0
##
```

```
sensitivity_initial <- conf_matrix_initial$sensitivity
sensitivity_optimized <- conf_matrix_optimized$sensitivity
```

Accuracy has increased from 0.661 to 0.763 sensitivity has changed from 0.406 to 0.844,Specificity by the above reasons,The model performance has been improved after using grid search.