Data622_HW2

Samriti Malhotra

12/1/2020

PART A

STEP#0: Pick any two classifiers of (SVM,Logistic,DecisionTree,NaiveBayes). Pick heart or ecoli dataset. Heart is simpler and ecoli compounds the problem as it is NOT a balanced dataset. From a grading perspective both carry the same weight. STEP#1 For each classifier, Set a seed (43) STEP#2 Do a 80/20 split and determine the Accuracy, AUC and as many metrics as returned by the Caret package (confusionMatrix) Call this the base_metric. Note down as best as you can development (engineering) cost as well as computing cost(elapsed time). Start with the original dataset and set a seed (43). Then run a cross validation of 5 and 10 of the model on the training set. Determine the same set of metrics and compare the cv_metrics with the base_metric. Note down as best as you can development (engineering) cost as well as computing cost(elapsed time). Start with the original dataset and set a seed (43) Then run a bootstrap of 200 resamples and compute the same set of metrics and for each of the two classifiers build a three column table for each experiment (base, bootstrap, cross-validated). Note down as best as you can development (engineering) cost as well as computing cost(elapsed time).

Read Data

```
##   age sex cp trestbps chol fbs restecg thalach exang oldpeak slope ca thal
## 1  63   1  3      145  233   1       0     150     0     2.3     0  0    1
## 2  37   1  2      130  250   0       1     187     0     3.5     0  0    2
## 3  41   0  1      130  204   0       0     172     0     1.4     2  0    2
## 4  56   1  1      120  236   0       1     178     0     0.8     2  0    2
## 5  57   0  0      120  354   0       1     163     1     0.6     2  0    2
## 6  57   1  0      140  192   0       1     148     0     0.4     1  0    1
##   target
## 1      1
## 2      1
## 3      1
## 4      1
## 5      1
## 6      1
```

```
## 'data.frame':    303 obs. of  14 variables:
##  $ age     : int  63 37 41 56 57 57 56 44 52 57 ...
##  $ sex     : int  1 1 0 1 0 1 0 1 1 1 ...
##  $ cp      : int  3 2 1 1 0 0 1 1 2 2 ...
##  $ trestbps: int  145 130 130 120 120 140 140 120 172 150 ...
##  $ chol    : int  233 250 204 236 354 192 294 263 199 168 ...
##  $ fbs     : int  1 0 0 0 0 0 0 0 1 0 ...
##  $ restecg : int  0 1 0 1 1 1 0 1 1 1 ...
##  $ thalach : int  150 187 172 178 163 148 153 173 162 174 ...
##  $ exang   : int  0 0 0 0 1 0 0 0 0 0 ...
##  $ oldpeak : num  2.3 3.5 1.4 0.8 0.6 0.4 1.3 0 0.5 1.6 ...
```

```
##  $ slope   : int  0 0 2 2 2 1 1 2 2 2 ...
##  $ ca      : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ thal    : int  1 2 2 2 2 1 2 3 3 2 ...
##  $ target  : int  1 1 1 1 1 1 1 1 1 1 ...
```

Data Transformation

Split Data

```
x <- floor(0.8 * nrow(heart))
raw <- sample(seq_len(nrow(heart)), size = x)
train_heart <- heart[ raw,]
test_heart  <- heart[-raw,]
```

Base Decision Tree

```
timer <- proc.time()

set.seed(43)

train_model = train(
    form = target ~ .,
    data = train_heart,
    trControl = trainControl(method="none"),
    method = "rpart"
    )
    print(train_model)
```

```
## CART
##
## 242 samples
##  13 predictor
##   2 classes: '0', '1'
##
## No pre-processing
## Resampling: None
```

```
dt_cm <- confusionMatrix(predict(train_model, subset(test_heart, select = -c(target))), test_heart$targ
print(paste("Base Decision Tree", 'Results'))
```

```
## [1] "Base Decision Tree Results"
```

```
    print(dt_cm)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0  0  0
##          1 33 28
##
##                Accuracy : 0.459
##                  95% CI : (0.3306, 0.5915)
##     No Information Rate : 0.541
##     P-Value [Acc > NIR] : 0.921
##
```

```
##                   Kappa : 0
##
##  Mcnemar's Test P-Value : 2.54e-08
##
##             Sensitivity : 0.000
##             Specificity : 1.000
##          Pos Pred Value :   NaN
##          Neg Pred Value : 0.459
##              Prevalence : 0.541
##          Detection Rate : 0.000
##    Detection Prevalence : 0.000
##       Balanced Accuracy : 0.500
##
##        'Positive' Class : 0
##
```

```
    duration <- (proc.time() - timer)[[3]]


    #metrics
    dt_accuracy <- dt_cm$overall[[1]]
    dt_auc <- as.numeric(auc(roc(test_heart$target, factor(predict(train_model, test_heart), ordered = T
    dt_sensitivity <- dt_cm$byClass[[1]]
    dt_specificity <- dt_cm$byClass[[2]]
    dt_precision <- dt_cm$byClass[[5]]
    dt_recall <- dt_cm$byClass[[6]]
    dt_f1_score <- dt_cm$byClass[[7]]
    dt_duration <- duration

    base_dt <- c("Decision Tree Base ",round(dt_accuracy,2),round(dt_auc,2),round(dt_sensitivity,2),roun
```

Base Support Vector Machine

```
train_model = train(
    form = target ~ .,
    data = train_heart,
    trControl = trainControl(method="none"),
    method = "svmLinear"
    )
    print(train_model)
```

```
## Support Vector Machines with Linear Kernel
##
## 242 samples
##  13 predictor
##   2 classes: '0', '1'
##
## No pre-processing
## Resampling: None
```

```
SVM_cm <- confusionMatrix(predict(train_model, subset(test_heart, select = -c(target))), test_heart$targ
print(paste("Base SVM", 'Results'))
```

```
## [1] "Base SVM Results"
```

```r
print(SVM_cm)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 24  5
##          1  9 23
##
##                Accuracy : 0.7705
##                  95% CI : (0.645, 0.8685)
##     No Information Rate : 0.541
##     P-Value [Acc > NIR] : 0.0001784
##
##                   Kappa : 0.5428
##
##  Mcnemar's Test P-Value : 0.4226781
##
##             Sensitivity : 0.7273
##             Specificity : 0.8214
##          Pos Pred Value : 0.8276
##          Neg Pred Value : 0.7188
##              Prevalence : 0.5410
##          Detection Rate : 0.3934
##    Detection Prevalence : 0.4754
##       Balanced Accuracy : 0.7744
##
##        'Positive' Class : 0
##
```

```r
duration <- (proc.time() - timer)[[3]]


#metrics
SVM_accuracy <- SVM_cm$overall[[1]]
SVM_auc <- as.numeric(auc(roc(test_heart$target, factor(predict(train_model, test_heart), ordered =
SVM_sensitivity <- SVM_cm$byClass[[1]]
SVM_specificity <- SVM_cm$byClass[[2]]
SVM_precision <- SVM_cm$byClass[[5]]
SVM_recall <- SVM_cm$byClass[[6]]
SVM_f1_score <- SVM_cm$byClass[[7]]
SVM_duration <- duration

base_svm <- c("SVM Base ",round(SVM_accuracy,2),round(SVM_auc,2),round(SVM_sensitivity,2),round(SVM
```

Cross Validation of 5 Decision Tree

```r
train_model = train(
  form = target ~ .,
  data = heart,
  trControl = trainControl(method = "cv", number = 5, savePredictions = 'final'),
  method = "rpart"
  )
```

```r
print(train_model)
```

```
## CART
##
## 303 samples
##  13 predictor
##   2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 243, 242, 242, 242, 243
## Resampling results across tuning parameters:
##
##   cp          Accuracy   Kappa
##   0.03623188  0.7592896  0.5144704
##   0.03985507  0.7494536  0.4959395
##   0.48550725  0.6495082  0.2575313
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was cp = 0.03623188.
```

```r
#confusion matrix
dt_5_cm <- confusionMatrix(train_model$pred[order(train_model$pred$rowIndex),]$pred, heart$target)

print(paste("Decision Tree 5 Fold", 'Results'))
```

```
## [1] "Decision Tree 5 Fold Results"
```

```r
print(dt_5_cm)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 100  35
##          1  38 130
##
##                Accuracy : 0.7591
##                  95% CI : (0.7069, 0.8061)
##     No Information Rate : 0.5446
##     P-Value [Acc > NIR] : 8.799e-15
##
##                   Kappa : 0.5134
##
##  Mcnemar's Test P-Value : 0.8149
##
##             Sensitivity : 0.7246
##             Specificity : 0.7879
##          Pos Pred Value : 0.7407
##          Neg Pred Value : 0.7738
##              Prevalence : 0.4554
##          Detection Rate : 0.3300
##    Detection Prevalence : 0.4455
##       Balanced Accuracy : 0.7563
##
```

```
##        'Positive' Class : 0
##
```

```
  duration <- (proc.time() - timer)[[3]]

  # metrics
  dt_5_cmaccuracy <- dt_5_cm$overall[[1]]
  dt_5_cmauc <- as.numeric(auc(roc(test_heart$target, factor(predict(train_model, test_heart), ordered =
  dt_5_cmsensitivity <- dt_5_cm$byClass[[1]]
  dt_5_cmspecificity <- dt_5_cm$byClass[[2]]
  dt_5_cmprecision <- dt_5_cm$byClass[[5]]
  dt_5_cmrecall <- dt_5_cm$byClass[[6]]
  dt_5_cmf1_score <- dt_5_cm$byClass[[7]]
  dt_5_duration <- duration

   dt_5 <- c("Decision Tree 5 fold",round(dt_5_cmaccuracy,2),round(dt_5_cmauc,2),round(dt_5_cmsensitivi
```

Cross Validation of 5 SVM

```
train_model = train(
  form = target ~ .,
  data = heart,
  trControl = trainControl(method = "cv", number = 5, savePredictions = 'final'),
  method = "svmLinear"
  )

  print(train_model)
```

```
## Support Vector Machines with Linear Kernel
##
## 303 samples
##  13 predictor
##   2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 242, 243, 243, 242, 242
## Resampling results:
##
##   Accuracy   Kappa
##   0.8314754  0.6574938
##
## Tuning parameter 'C' was held constant at a value of 1
```

```
  #confusion matrix
  SVM_5_cm <- confusionMatrix(train_model$pred[order(train_model$pred$rowIndex),]$pred, heart$target)

  print(paste("SVM 5 Fold", 'Results'))
```

```
## [1] "SVM 5 Fold Results"
```

```
  print(SVM_5_cm)
```

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

```
## Prediction   0   1
##          0 107  20
##          1  31 145
##
##                Accuracy : 0.8317
##                  95% CI : (0.7847, 0.872)
##     No Information Rate : 0.5446
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.6584
##
##  Mcnemar's Test P-Value : 0.1614
##
##             Sensitivity : 0.7754
##             Specificity : 0.8788
##          Pos Pred Value : 0.8425
##          Neg Pred Value : 0.8239
##              Prevalence : 0.4554
##          Detection Rate : 0.3531
##    Detection Prevalence : 0.4191
##       Balanced Accuracy : 0.8271
##
##        'Positive' Class : 0
##
```

```r
  duration <- (proc.time() - timer)[[3]]

  # metrics
  SVM_5_cmaccuracy <- SVM_5_cm$overall[[1]]
  SVM_5_cmauc <- as.numeric(auc(roc(test_heart$target, factor(predict(train_model, test_heart), ordered
  SVM_5_cmsensitivity <- SVM_5_cm$byClass[[1]]
  SVM_5_cmspecificity <- SVM_5_cm$byClass[[2]]
  SVM_5_cmprecision <- SVM_5_cm$byClass[[5]]
  SVM_5_cmrecall <- SVM_5_cm$byClass[[6]]
  SVM_5_cmf1_score <- SVM_5_cm$byClass[[7]]
  SVM_5_duration <- duration

  SVM_5 <- c("SVM 5 Fold ",round(SVM_5_cmaccuracy,2),round(SVM_5_cmauc,2),round(SVM_5_cmsensitivity,2),
```

Cross Validation of 10 Decision Tree

```r
train_model = train(
  form = target ~ .,
  data = heart,
  trControl = trainControl(method = "cv", number = 10, savePredictions = 'final'),
  method = "rpart"
  )

  print(train_model)
```

```
## CART
##
## 303 samples
##  13 predictor
```

```
##   2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 273, 273, 273, 273, 272, 272, ...
## Resampling results across tuning parameters:
##
##   cp          Accuracy   Kappa
##   0.03623188  0.7425806  0.4835713
##   0.03985507  0.7654839  0.5274404
##   0.48550725  0.6007527  0.1532926
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was cp = 0.03985507.
```

```
#confusion matrix
dt_10_cm <- confusionMatrix(train_model$pred[order(train_model$pred$rowIndex),]$pred, heart$target)

print(paste("Decision Tree 10 Fold", 'Results'))
```

```
## [1] "Decision Tree 10 Fold Results"
```

```
print(dt_10_cm)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 101  34
##          1  37 131
##
##                Accuracy : 0.7657
##                  95% CI : (0.7139, 0.8122)
##     No Information Rate : 0.5446
##     P-Value [Acc > NIR] : 1.208e-15
##
##                   Kappa : 0.5268
##
##  Mcnemar's Test P-Value : 0.8124
##
##             Sensitivity : 0.7319
##             Specificity : 0.7939
##          Pos Pred Value : 0.7481
##          Neg Pred Value : 0.7798
##              Prevalence : 0.4554
##          Detection Rate : 0.3333
##    Detection Prevalence : 0.4455
##       Balanced Accuracy : 0.7629
##
##        'Positive' Class : 0
##
```

```
duration <- (proc.time() - timer)[[3]]

# metrics
dt_10_cmaccuracy <- dt_10_cm$overall[[1]]
```

```r
  dt_10_cmauc <- as.numeric(auc(roc(test_heart$target, factor(predict(train_model, test_heart), ordered
  dt_10_cmsensitivity <- dt_10_cm$byClass[[1]]
  dt_10_cmspecificity <- dt_10_cm$byClass[[2]]
  dt_10_cmprecision <- dt_10_cm$byClass[[5]]
  dt_10_cmrecall <- dt_10_cm$byClass[[6]]
  dt_10_cmf1_score <- dt_10_cm$byClass[[7]]
  dt_10_duration <- duration

   dt_10 <- c("Decision Tree 10 Fold",round(dt_10_cmaccuracy,2),round(dt_10_cmauc,2),round(dt_10_cmsens
```

Cross Validation of 10 SVM

```r
train_model = train(
  form = target ~ .,
  data = heart,
  trControl = trainControl(method = "cv", number = 10, savePredictions = 'final'),
  method = "svmLinear"
  )

  print(train_model)
```

```
## Support Vector Machines with Linear Kernel
##
## 303 samples
##  13 predictor
##   2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 272, 274, 273, 272, 273, 273, ...
## Resampling results:
##
##   Accuracy  Kappa
##   0.83835   0.6714231
##
## Tuning parameter 'C' was held constant at a value of 1
```

```r
  #confusion matrix
  SVM_10_cm <- confusionMatrix(train_model$pred[order(train_model$pred$rowIndex),]$pred, heart$target)

  print(paste("SVM 10 Fold", 'Results'))
```

```
## [1] "SVM 10 Fold Results"
```

```r
  print(SVM_10_cm)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 108  19
##          1  30 146
##
##                Accuracy : 0.8383
##                  95% CI : (0.7919, 0.8779)
```

```
##     No Information Rate : 0.5446
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.6718
##
##  Mcnemar's Test P-Value : 0.1531
##
##             Sensitivity : 0.7826
##             Specificity : 0.8848
##          Pos Pred Value : 0.8504
##          Neg Pred Value : 0.8295
##              Prevalence : 0.4554
##          Detection Rate : 0.3564
##    Detection Prevalence : 0.4191
##       Balanced Accuracy : 0.8337
##
##        'Positive' Class : 0
##
```

```
  duration <- (proc.time() - timer)[[3]]

  # metrics
  SVM_10_cmaccuracy <- SVM_10_cm$overall[[1]]
  SVM_10_cmauc <- as.numeric(auc(roc(test_heart$target, factor(predict(train_model, test_heart), ordere
  SVM_10_cmsensitivity <- SVM_10_cm$byClass[[1]]
  SVM_10_cmspecificity <- SVM_10_cm$byClass[[2]]
  SVM_10_cmprecision <- SVM_10_cm$byClass[[5]]
  SVM_10_cmrecall <- SVM_10_cm$byClass[[6]]
  SVM_10_cmf1_score <- SVM_10_cm$byClass[[7]]
  SVM_10_duration <- duration

  SVM_10 <- c("SNM 10 Fold",round(SVM_10_cmaccuracy,2),round(SVM_10_cmauc,2),round(SVM_10_cmsensitivity
```

Bootstrap of 200 Resamples | Decision Tree

```
train_model = train(
    form = target ~ .,
    data = heart,
    trControl = trainControl(method="boot", number=200, savePredictions = 'final', returnResamp = 'fina
    method = "rpart"
    )

    duration <- (proc.time() - timer)[[3]]

    accuracy <- c()
    auc <- c()
    sensitivity <- c()
    specificity <- c()
    precision <- c()
    recall <- c()
    f1_score <- c()
    i <- 1

    pred_df <- train_model$pred
```

```
    for (resample in unique(pred_df$Resample)){
      temp <- filter(pred_df, Resample == resample)
      model_cm <- confusionMatrix(temp$pred, temp$obs)
      accuracy[i] <- model_cm$overall[[1]]
      auc[[i]] <- auc(roc(as.numeric(temp$pred, ordered = TRUE), as.numeric(temp$obs, ordered = TRUE)))
      sensitivity[[i]] <- model_cm$byClass[[1]]
      specificity[[i]] <- model_cm$byClass[[2]]
      precision[[i]] <- model_cm$byClass[[5]]
      recall[[i]] <- model_cm$byClass[[6]]
      f1_score[[i]] <- model_cm$byClass[[7]]
      i <- i + 1
    }

    dt_200_accuracy <- mean(accuracy)
    dt_200_auc <- mean(auc)
    dt_200_sensitivity <- mean(sensitivity)
    dt_200_specificity <- mean(specificity)
    dt_200_precision <- mean(precision)
    dt_200_recall <- mean(recall)
    dt_200_f1_score <- mean(f1_score)
    dt_200_duration <- duration

    dt_200 <- c("Decision Tree Bootstrap",round(dt_200_accuracy,2),round(dt_200_auc,2),round(dt_200_sens
```

Bootstrap of 200 Resamples | SVM

```
train_model = train(
    form = target ~ .,
    data = heart,
    trControl = trainControl(method="boot", number=200, savePredictions = 'final', returnResamp = 'fina
    method = "svmLinear"
    )

    duration <- (proc.time() - timer)[[3]]

    accuracy <- c()
    auc <- c()
    sensitivity <- c()
    specificity <- c()
    precision <- c()
    recall <- c()
    f1_score <- c()
    i <- 1

    pred_df <- train_model$pred
    for (resample in unique(pred_df$Resample)){
      temp <- filter(pred_df, Resample == resample)
      model_cm <- confusionMatrix(temp$pred, temp$obs)
      accuracy[i] <- model_cm$overall[[1]]
      auc[[i]] <- auc(roc(as.numeric(temp$pred, ordered = TRUE), as.numeric(temp$obs, ordered = TRUE)))
      sensitivity[[i]] <- model_cm$byClass[[1]]
      specificity[[i]] <- model_cm$byClass[[2]]
      precision[[i]] <- model_cm$byClass[[5]]
```

```r
      recall[[i]] <- model_cm$byClass[[6]]
      f1_score[[i]] <- model_cm$byClass[[7]]
      i <- i + 1
    }

    SVM_200_accuracy <- mean(accuracy)
    SVM_200_auc <- mean(auc)
    SVM_200_sensitivity <- mean(sensitivity)
    SVM_200_specificity <- mean(specificity)
    SVM_200_precision <- mean(precision)
    SVM_200_recall <- mean(recall)
    SVM_200_f1_score <- mean(f1_score)
    SVM_200_duration <- duration

    SVM_200 <- c("SVM  Bootstrap",round(SVM_200_accuracy,2),round(SVM_200_auc,2),round(SVM_200_sensitivi
```

```r
results <- data.frame(matrix(ncol = 10, nrow = 0))
results <- rbind(results,base_dt,dt_5, dt_10  , dt_200 , base_svm , SVM_5, SVM_10 , SVM_200)
colnames(results) <- c("Model","Accuracy", "AUC","Sensitivity", "Specificity", "Precision", "Recall", "F
#results
kable(results) %>%
  kable_styling(bootstrap_options = "bordered") %>%
  row_spec(0, bold = T, color = "black", background = "#7fcdcc")
```

Model	Accuracy	AUC	Sensitivity	Specificity	Precision	Recall	F1_Score	Durat
Decision Tree Base	0.46	0.5	0	1	NA	0	NA	0.55
Decision Tree 5 fold	0.76	0.81	0.72	0.79	0.74	0.72	0.73	2.33
Decision Tree 10 Fold	0.77	0.79	0.73	0.79	0.75	0.73	0.74	4.56
Decision Tree Bootstrap	0.74	0.74	0.69	0.79	0.73	0.69	0.71	9.63
SVM Base	0.77	0.77	0.73	0.82	0.83	0.73	0.77	1.61
SVM 5 Fold	0.83	0.89	0.78	0.88	0.84	0.78	0.81	3.67
SNM 10 Fold	0.84	0.89	0.78	0.88	0.85	0.78	0.82	5.4
SVM Bootstrap	0.82	0.82	0.78	0.85	0.81	0.78	0.79	19.2

PART B

For the same dataset, set seed (43) split 80/20. Using randomForest grow three different forests varuing the number of trees atleast three times. Start with seeding and fresh split for each forest. Note down as best as you can development (engineering) cost as well as computing cost(elapsed time) for each run. And compare these results with the experiment in Part A. Submit a pdf and executable script in python or R.

```r
set.seed(43)
x <- floor(0.8 * nrow(heart))
raw <- sample(seq_len(nrow(heart)), size = x)
train_heart <- heart[ raw,]
test_heart  <- heart[-raw,]
```

12 Trees

```r
    train_model = train(
    form = target ~ .,
    data = train_heart,
    trControl = trainControl(),
```

```r
    ntree = 12,
    method = "rf"
    )
    print(train_model)
```

```
## Random Forest
##
## 242 samples
##  13 predictor
##   2 classes: '0', '1'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 242, 242, 242, 242, 242, 242, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.7960814  0.5838828
##   12    0.7664892  0.5275995
##   22    0.7653989  0.5217815
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 2.
```

```r
 # confusion Matrix from caret

    rf_cm <- confusionMatrix(predict(train_model, subset(test_heart, select = -c(target))), test_heart$t

    print(paste("RF 12", 'Results'))
```

```
## [1] "RF 12 Results"
```

```r
    print(rf_cm)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 23  2
##          1  8 28
##
##                Accuracy : 0.8361
##                  95% CI : (0.7191, 0.9185)
##     No Information Rate : 0.5082
##     P-Value [Acc > NIR] : 9.418e-08
##
##                   Kappa : 0.6731
##
##  Mcnemar's Test P-Value : 0.1138
##
##             Sensitivity : 0.7419
##             Specificity : 0.9333
##          Pos Pred Value : 0.9200
##          Neg Pred Value : 0.7778
##              Prevalence : 0.5082
```

```
##          Detection Rate : 0.3770
##    Detection Prevalence : 0.4098
##       Balanced Accuracy : 0.8376
##
##        'Positive' Class : 0
##
```

```
    rf_duration <- (proc.time() - timer)[[3]]

    # metrics
    rf_accuracy <- rf_cm$overall[[1]]
    rf_auc <- as.numeric(auc(roc(test_heart$target, factor(predict(train_model, test_heart), ordered = '
```

```
## Setting levels: control = 0, case = 1
```

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

```
    rf_sensitivity <- rf_cm$byClass[[1]]
    rf_specificity <- rf_cm$byClass[[2]]
    rf_precision <- rf_cm$byClass[[5]]
    rf_recall <- rf_cm$byClass[[6]]
    rf_f1_score <- rf_cm$byClass[[7]]
    rf_duration <- rf_duration

     rf_12 <- c("Random Forrest 16",round(rf_accuracy,2),round(rf_auc,2),round(rf_sensitivity,2),round(
```

24 Trees

```
    train_model = train(
    form = target ~ .,
    data = train_heart,
    trControl = trainControl(),
    ntree = 24,
    method = "rf"
    )
    print(train_model)
```

```
## Random Forest
##
## 242 samples
##  13 predictor
##   2 classes: '0', '1'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 242, 242, 242, 242, 242, 242, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.7889507  0.5704351
##   12    0.7725683  0.5393684
##   22    0.7655967  0.5255263
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 2.
```

```r
# confusion Matrix from caret

    rf_24_cm <- confusionMatrix(predict(train_model, subset(test_heart, select = -c(target))), test_hear

    print(paste("RF 24", 'Results'))
```

```
## [1] "RF 24 Results"
```

```r
    print(rf_cm)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 23  2
##          1  8 28
##
##                Accuracy : 0.8361
##                  95% CI : (0.7191, 0.9185)
##     No Information Rate : 0.5082
##     P-Value [Acc > NIR] : 9.418e-08
##
##                   Kappa : 0.6731
##
##  Mcnemar's Test P-Value : 0.1138
##
##             Sensitivity : 0.7419
##             Specificity : 0.9333
##          Pos Pred Value : 0.9200
##          Neg Pred Value : 0.7778
##              Prevalence : 0.5082
##          Detection Rate : 0.3770
##    Detection Prevalence : 0.4098
##       Balanced Accuracy : 0.8376
##
##        'Positive' Class : 0
##
```

```r
    rf_24_duration <- (proc.time() - timer)[[3]]

    # metrics
    rf_24_accuracy <- rf_24_cm$overall[[1]]
    rf_24_auc <- as.numeric(auc(roc(test_heart$target, factor(predict(train_model, test_heart), ordered
```

```
## Setting levels: control = 0, case = 1
```

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

```r
    rf_24_sensitivity <- rf_24_cm$byClass[[1]]
    rf_24_specificity <- rf_24_cm$byClass[[2]]
    rf_24_precision <- rf_24_cm$byClass[[5]]
    rf_24_recall <- rf_24_cm$byClass[[6]]
    rf_24_f1_score <- rf_24_cm$byClass[[7]]
    rf_24_duration <- rf_24_duration

     rf_24 <- c("Random Forrest 24",round(rf_24_accuracy,2),round(rf_24_auc,2),round(rf_24_sensitivity,
```

114 Trees

```r
train_model = train(
form = target ~ .,
data = train_heart,
trControl = trainControl(),
ntree = 114,
method = "rf"
)
print(train_model)
```

```
## Random Forest
##
## 242 samples
##  13 predictor
##   2 classes: '0', '1'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 242, 242, 242, 242, 242, 242, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.7964888  0.5842318
##   12    0.7679865  0.5267527
##   22    0.7650865  0.5217642
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 2.
```

```r
# confusion Matrix from caret

rf_114_cm <- confusionMatrix(predict(train_model, subset(test_heart, select = -c(target))), test_hea

print(paste("RF 114", 'Results'))
```

```
## [1] "RF 114 Results"
```

```r
print(rf_cm)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 23  2
##          1  8 28
##
##                Accuracy : 0.8361
##                  95% CI : (0.7191, 0.9185)
##     No Information Rate : 0.5082
##     P-Value [Acc > NIR] : 9.418e-08
##
##                   Kappa : 0.6731
##
##  Mcnemar's Test P-Value : 0.1138
##
```

```
##              Sensitivity : 0.7419
##              Specificity : 0.9333
##           Pos Pred Value : 0.9200
##           Neg Pred Value : 0.7778
##               Prevalence : 0.5082
##           Detection Rate : 0.3770
##     Detection Prevalence : 0.4098
##        Balanced Accuracy : 0.8376
##
##         'Positive' Class : 0
##
```

```
    rf_114_duration <- (proc.time() - timer)[[3]]

    # metrics
    rf_114_accuracy <- rf_114_cm$overall[[1]]
    rf_114_auc <- as.numeric(auc(roc(test_heart$target, factor(predict(train_model, test_heart), ordere
```

```
## Setting levels: control = 0, case = 1
```

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

```
    rf_114_sensitivity <- rf_114_cm$byClass[[1]]
    rf_114_specificity <- rf_114_cm$byClass[[2]]
    rf_114_precision <- rf_114_cm$byClass[[5]]
    rf_114_recall <- rf_114_cm$byClass[[6]]
    rf_114_f1_score <- rf_114_cm$byClass[[7]]
    rf_114_duration <- rf_114_duration

     rf_114 <- c("Random Forrest 114",round(rf_114_accuracy,2),round(rf_114_auc,2),round(rf_114_sensiti
```

Part C

Include a summary of your findings. Which of the two methods bootstrap vs cv do you recommend to your customer? And why? Be elaborate. Including computing costs, engineering costs and model performance. Did you incorporate Pareto's maxim or the Razor and how did these two heuristics influence your decision?

```
final_results <- data.frame(matrix(ncol = 10, nrow = 0))
final_results <- rbind(final_results,base_dt,dt_5, dt_10  , dt_200 , base_svm , SVM_5, SVM_10 , SVM_200
colnames(final_results) <- c("Model","Accuracy", "AUC","Sensitivity", "Specificity", "Precision", "Recal

kable(final_results) %>%
  kable_styling(bootstrap_options = "bordered") %>%
  row_spec(0, bold = T, color = "black", background = "#7fcdcc")
```

Model	Accuracy	AUC	Sensitivity	Specificity	Precision	Recall	F1_Score	Durat
Decision Tree Base	0.46	0.5	0	1	NA	0	NA	0.55
Decision Tree 5 fold	0.76	0.81	0.72	0.79	0.74	0.72	0.73	2.33
Decision Tree 10 Fold	0.77	0.79	0.73	0.79	0.75	0.73	0.74	4.56
Decision Tree Bootstrap	0.74	0.74	0.69	0.79	0.73	0.69	0.71	9.63
SVM Base	0.77	0.77	0.73	0.82	0.83	0.73	0.77	1.61
SVM 5 Fold	0.83	0.89	0.78	0.88	0.84	0.78	0.81	3.67
SNM 10 Fold	0.84	0.89	0.78	0.88	0.85	0.78	0.82	5.4
SVM Bootstrap	0.82	0.82	0.78	0.85	0.81	0.78	0.79	19.2
Random Forrest 16	0.84	0.85	0.74	0.93	0.92	0.74	0.82	22.25
Random Forrest 24	0.8	0.8	0.77	0.83	0.83	0.77	0.8	23.72
Random Forrest 114	0.82	0.82	0.77	0.87	0.86	0.77	0.81	27.09

Conclusion

While comparing bootstrap and cross validation. Cross validation has better performance metrics and less computational time. SVM had better model performance but elapsed time is higher than the desicion tree. 10 fold produced better results. So 10 fold cross validation should be used for both classifiers. As per Occam's Razor the problem-solving principle that "entities should not be multiplied without necessity", or more simply, the simplest explanation is usually the right one.

Pareto's maxim or the Razor