Data 622 - Homework 2

Salma Elshahawy

11/4/2020

Contents

Data 621 Homework 4

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).

```r
#Load libraries
library(tidyverse)
library(kableExtra)
library(rsample)
library(recipes)
library(parsnip)
library(yardstick)
library(viridisLite)
library(GGally)
library(tibble)
```

```r
heart.data <- read.csv("https://archive.ics.uci.edu/ml/machine-learning-databases/heart-disease/processe
names(heart.data) <- c( "age", "sex", "cp", "trestbps", "chol","fbs", "restecg",
                   "thalach","exang", "oldpeak","slope", "ca", "thal", "num")
```

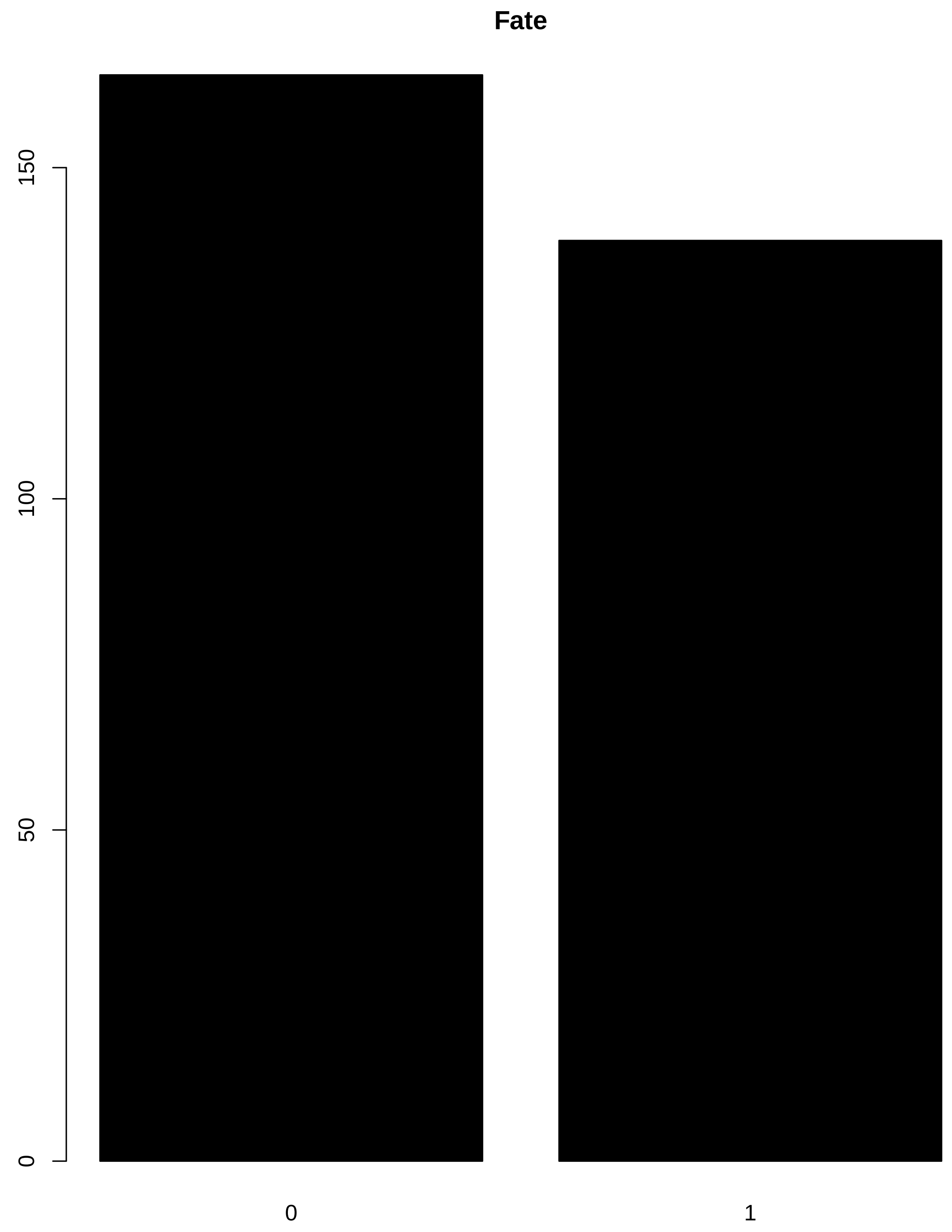
```r
head(heart.data,3)
```

```
  age sex cp trestbps chol fbs restecg thalach exang oldpeak slope ca thal num
1  63   1  1      145  233   1       2     150     0     2.3     3  0    6   0
2  67   1  4      160  286   0       2     108     1     1.5     2  3    3   2
3  67   1  4      120  229   0       2     129     1     2.6     2  2    7   1
```

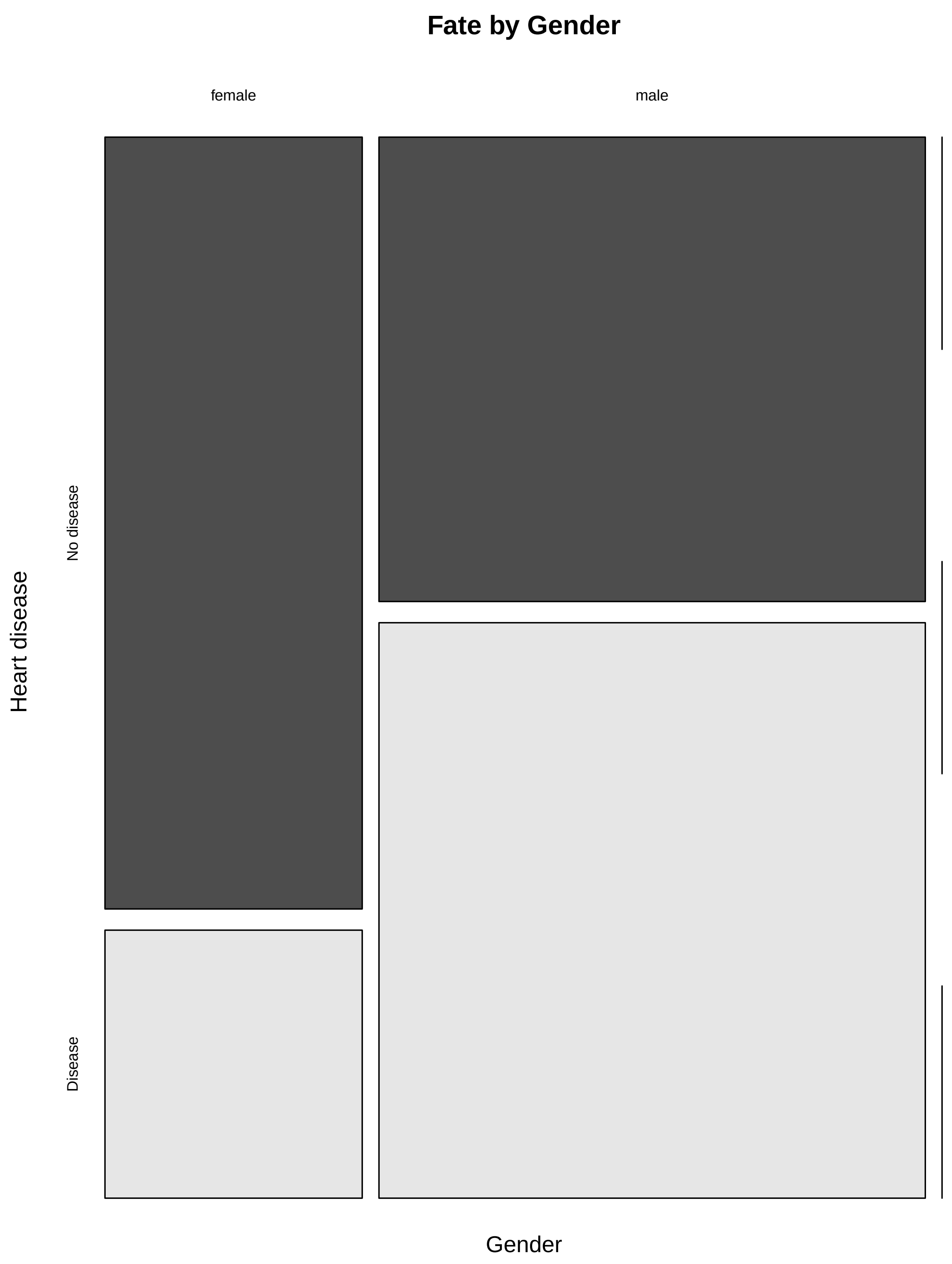
```r
dim(heart.data)
```

```
[1] 303  14
```

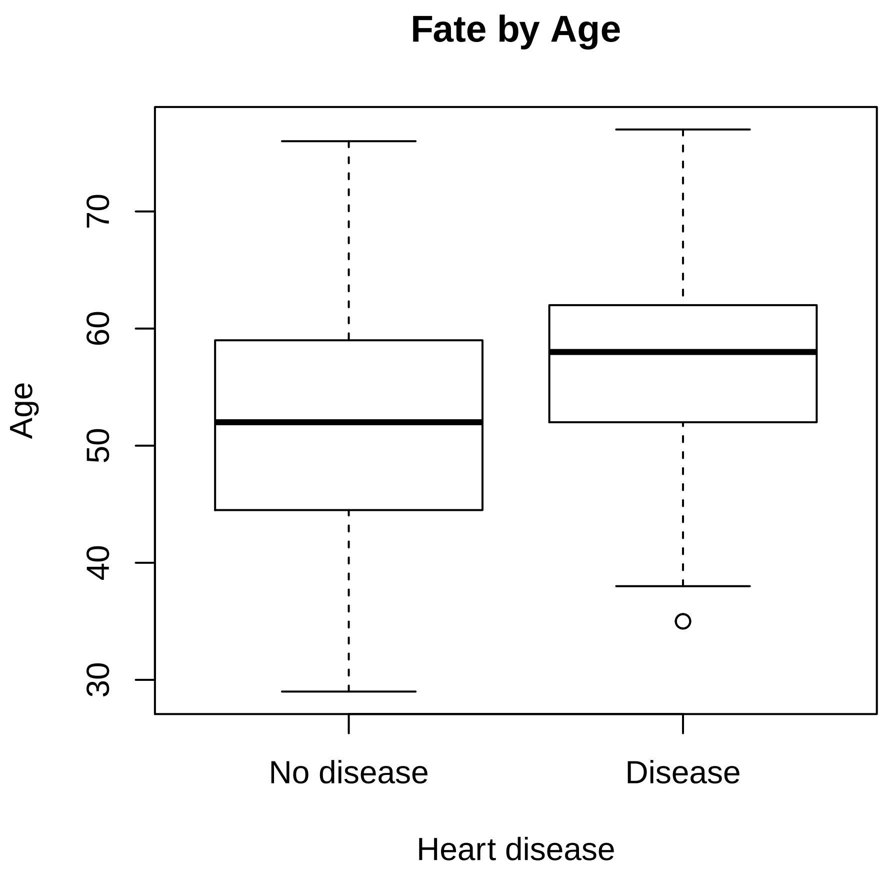
```r
heart.data$num[heart.data$num > 0] <- 1
barplot(table(heart.data$num),
        main="Fate", col="black")
```



```
#Function needed to convert classes of predictor values
convert.magic <- function(obj,types){
    for (i in 1:length(obj)){
        FUN <- switch(types[i],character = as.character,
                                   numeric = as.numeric,
                                   factor = as.factor)
        obj[,i] <- FUN(obj[,i])
    }
    obj
}
convert.names <- function(row){
  row=gsub("sex1", "male", row)
  row=gsub("thal7", "reversable defect thalassemia", row)
  row=gsub("thal6", "fixed defect thalassemia", row)
  row=gsub("cp4", "asymptomatic chest pain", row)
  row=gsub("cp3", "non-anginal chest pain", row)
  row=gsub("cp2", "atypical angina chest pain", row)
  row=gsub("oldpeak", "ST depression from exercise", row)
  row=gsub("thalach", "maximum heart rate achieved", row)
  row=gsub("trestbps", "resting blood pressure", row)
  row=gsub("ca2", "2 major vessels col/b fluoro., ca2", row)
  row=gsub("ca1", "1 major vessel col/b fluoro., ca1", row)
  row=gsub("slope2", "flat peak exercise ST segment", row)
  row=gsub("slope1", "upsloping peak exercise ST segment", row)
  row=gsub("slope3", "downsloping peak exercise ST segment", row)
  row=gsub("chol", "serum cholestoral", row)
  row=gsub("exang", "exercise induced angina", row)
  row=gsub("restecg2", "restec: showing left ventricular hypertrophy
                      by Estes criteria", row)
  row=gsub("restecg1", "restec: having ST-T wave abnormality", row)
  row=gsub("fbs1", "fasting blood sugar > 120 mg/dl", row)
}

# change a few predictor variables from integer to factors (make dummies)
chclass <-c("numeric","factor","factor","numeric","numeric","factor","factor","numeric","factor","numer

heart.data <- convert.magic(heart.data,chclass)

heart = heart.data #add labels only for plot
levels(heart$num) = c("No disease","Disease")
levels(heart$sex) = c("female","male","")
mosaicplot(heart$sex ~ heart$num,
           main="Fate by Gender", shade=FALSE,color=TRUE,
           xlab="Gender", ylab="Heart disease")
```



```r
boxplot(heart$age ~ heart$num,
        main="Fate by Age",
         ylab="Age",xlab="Heart disease")
```



```r
s = sum(is.na(heart.data))
heart.data <- na.omit(heart.data)
```

```r
library(caret)
set.seed(43)
inTrainRows <- createDataPartition(heart.data$num,p=0.8,list=FALSE)
trainData <- heart.data[inTrainRows,]
testData <-  heart.data[-inTrainRows,]
nrow(trainData)/(nrow(testData)+nrow(trainData)) #checking whether really 80% -> OK
```

```
[1] 0.8013468
```

Model

Base Model - logistic regression

```
set.seed(43)
log_base_a = Sys.time()
logRegModel <- train(num ~ ., data=trainData, method = 'glm', family = 'binomial')

logRegPrediction <- predict(logRegModel, testData)

logRegPredictionprob <- predict(logRegModel, testData, type='prob')[2]

(logRegConfMat <- caret::confusionMatrix(logRegPrediction, testData[,"num"]))
```

```
Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0 28  5
         1  4 22

               Accuracy : 0.8475
                 95% CI : (0.7301, 0.9278)
    No Information Rate : 0.5424
    P-Value [Acc > NIR] : 7.195e-07

                  Kappa : 0.6918

 Mcnemar's Test P-Value : 1

            Sensitivity : 0.8750
            Specificity : 0.8148
         Pos Pred Value : 0.8485
         Neg Pred Value : 0.8462
             Prevalence : 0.5424
         Detection Rate : 0.4746
   Detection Prevalence : 0.5593
      Balanced Accuracy : 0.8449

       'Positive' Class : 0
```

```
log_base_b = Sys.time()
paste0(round(as.numeric(difftime(time1 = log_base_b, time2 = log_base_a, units = "secs")), 3), " Elapse
```

```
[1] "2.859 Elapsed time in Seconds"
```

```
#ROC Curve
# library(pROC)
#
# (AUC$logReg <- roc(as.numeric(testData$num),as.numeric(as.matrix((logRegPredictionprob))))$auc)
# (Accuracy$logReg <- logRegConfMat$overall['Accuracy'])  #found names with str(logRegConfMat)
```

Base Model - SVM

```r
set.seed(43)
svm_base_a = Sys.time()
# for this to work add names to all levels (numbers not allowed)
feature.names=names(heart.data)

for (f in feature.names) {
  if (class(heart.data[[f]])=="factor") {
    levels <- unique(c(heart.data[[f]]))
    heart.data[[f]] <- factor(heart.data[[f]],
                   labels=make.names(levels))
  }
}

inTrainRows <- createDataPartition(heart.data$num,p=0.8,list=FALSE)
trainData2 <- heart.data[inTrainRows,]
testData2 <-  heart.data[-inTrainRows,]

fitControl <- trainControl(method = "none",
                           ## Estimate class probabilities
                           classProbs = TRUE,
                           summaryFunction = twoClassSummary)

svmModel <- train(num ~ ., data = trainData2,
                 method = "svmRadial",
                 trControl = fitControl,
                 preProcess = c("center", "scale"),
                 metric = "sens")

svmPrediction <- predict(svmModel, testData2)
svmPredictionprob <- predict(svmModel, testData2, type='prob')[2]
(svmConfMat <- caret::confusionMatrix(svmPrediction, testData2[,"num"]))
```

```
Confusion Matrix and Statistics

          Reference
Prediction X1 X2
        X1 26  7
        X2  6 20

               Accuracy : 0.7797
                 95% CI : (0.6527, 0.8771)
    No Information Rate : 0.5424
    P-Value [Acc > NIR] : 0.0001366

                  Kappa : 0.5548

 Mcnemar's Test P-Value : 1.0000000

            Sensitivity : 0.8125
            Specificity : 0.7407
         Pos Pred Value : 0.7879
         Neg Pred Value : 0.7692
             Prevalence : 0.5424
         Detection Rate : 0.4407
   Detection Prevalence : 0.5593
      Balanced Accuracy : 0.7766

       'Positive' Class : X1
```

```
svm_base_b = Sys.time()
paste0(round(as.numeric(difftime(time1 = svm_base_b, time2 = svm_base_a, units = "secs")), 3), " Elapse
```

```
[1] "2.967 Elapsed time in Seconds"
```

```
row.names_1 <-  c('Basic Logistic Regression',
               'basic SVM')
acc_1 <- c(
         round(logRegConfMat$overall['Accuracy'], 3),
         round(svmConfMat$overall['Accuracy'], 3))

sens_1 <- c(
         round(logRegConfMat$byClass['Sensitivity'], 3),
         round(svmConfMat$byClass['Sensitivity'], 3))

spec_1 <- c(
         round(logRegConfMat$byClass['Specificity'], 3),
         round(svmConfMat$byClass['Specificity'], 3))

kappa_1 <- c(
         round(logRegConfMat$overall['Kappa'], 3),
         round(svmConfMat$overall['Kappa'], 3))
F1_1 <- c(
         round(logRegConfMat$byClass['F1'], 3),
         round(svmConfMat$byClass['F1'], 3))

sum_df_basic <- tibble(Model = row.names_1,
                 Accuracy = acc_1,
                 Sensitivity = sens_1,
                 Specificity = spec_1,
                 Kappa = kappa_1,
                 F1 = F1_1)
sum_df_basic
```

```
# A tibble: 2 x 6
  Model                     Accuracy Sensitivity Specificity Kappa    F1
  <chr>                        <dbl>       <dbl>       <dbl> <dbl> <dbl>
1 Basic Logistic Regression    0.847       0.875       0.815 0.692 0.862
2 basic SVM                    0.78        0.812       0.741 0.555 0.8
```

The best model is the relative simple logistic regression model with an Area under the kappa of 0.862. We can predict heart disease with an accuracy of 0.847. The Sensitivity is 0.875 and the Specificity 0.815.

Using Cross validation - Logistic regression

```r
set.seed(43)
log_cv_a = Sys.time()
# for this to work add names to all levels (numbers not allowed)
feature.names=names(heart.data)

for (f in feature.names) {
  if (class(heart.data[[f]])=="factor") {
    levels <- unique(c(heart.data[[f]]))
    heart.data[[f]] <- factor(heart.data[[f]],
                   labels=make.names(levels))
  }
}

inTrainRows <- createDataPartition(heart.data$num,p=0.8,list=FALSE)
trainData2 <- heart.data[inTrainRows,]
testData2 <-  heart.data[-inTrainRows,]

# Kfold control
ctr_cv <- trainControl(method = "repeatedcv",
                           number = 10,
                           repeats =10,
                           classProbs = TRUE,
                           summaryFunction = twoClassSummary)

# k-fold training
log_cv <- train(num ~ ., data = trainData2,
                 method = "glm",
                 family = "binomial",
                 trControl = ctr_cv,
                 preProcess = c("center", "scale"),
                 metric = "ROC")

log_cv_pred <- predict(log_cv, testData2)
log_cv_Pred_prob <- predict(log_cv, testData2, type='prob')[2]
(logCvMat <- caret::confusionMatrix(log_cv_pred, testData2[,"num"]))
```

```
Confusion Matrix and Statistics

          Reference
Prediction X1 X2
        X1 28  5
        X2  4 22

               Accuracy : 0.8475
                 95% CI : (0.7301, 0.9278)
    No Information Rate : 0.5424
    P-Value [Acc > NIR] : 7.195e-07

                  Kappa : 0.6918

 Mcnemar's Test P-Value : 1
```

```
            Sensitivity : 0.8750
            Specificity : 0.8148
         Pos Pred Value : 0.8485
         Neg Pred Value : 0.8462
             Prevalence : 0.5424
         Detection Rate : 0.4746
   Detection Prevalence : 0.5593
      Balanced Accuracy : 0.8449

       'Positive' Class : X1
```

```
log_cv_b = Sys.time()
#ROC Curve
# AUC$svm <- roc(as.numeric(testData2$num),as.numeric(as.matrix((log_cv_Pred_prob))))$auc
# Accuracy$svm <- svmConfMat$overall['Accuracy']
paste0(round(as.numeric(difftime(time1 = log_cv_b, time2 = log_cv_a, units = "secs")), 3), " Elapsed ti
```

```
[1] "2.986 Elapsed time in Seconds"
```

Using Cross validation - SVM

```r
set.seed(43)
svm_cv_a = Sys.time()
# for this to work add names to all levels (numbers not allowed)
feature.names=names(heart.data)

for (f in feature.names) {
  if (class(heart.data[[f]])=="factor") {
    levels <- unique(c(heart.data[[f]]))
    heart.data[[f]] <- factor(heart.data[[f]],
                   labels=make.names(levels))
  }
}

inTrainRows <- createDataPartition(heart.data$num,p=0.8,list=FALSE)
trainData2 <- heart.data[inTrainRows,]
testData2 <-  heart.data[-inTrainRows,]

fitControl_2 <- trainControl(method = "repeatedcv",
                           ## Estimate class probabilities
                           classProbs = TRUE,
                           number = 10,
                           repeats = 10,
                           summaryFunction = twoClassSummary)

svmModel_2 <- train(num ~ ., data = trainData2,
                 method = "svmRadial",
                 trControl = fitControl_2,
                 preProcess = c("center", "scale"),
                 metric = "ROC")

svmPrediction_2 <- predict(svmModel_2, testData2)
svmPredictionprob_2 <- predict(svmModel_2, testData2, type='prob')[2]
(svmConfMat_2 <- caret::confusionMatrix(svmPrediction_2, testData2[,"num"]))
```

```
Confusion Matrix and Statistics

          Reference
Prediction X1 X2
        X1 26  4
        X2  6 23

               Accuracy : 0.8305
                 95% CI : (0.7103, 0.9156)
    No Information Rate : 0.5424
    P-Value [Acc > NIR] : 3.14e-06

                  Kappa : 0.6605

 Mcnemar's Test P-Value : 0.7518

            Sensitivity : 0.8125
            Specificity : 0.8519
         Pos Pred Value : 0.8667
         Neg Pred Value : 0.7931
             Prevalence : 0.5424
         Detection Rate : 0.4407
   Detection Prevalence : 0.5085
      Balanced Accuracy : 0.8322

       'Positive' Class : X1
```

```
svm_cv_b = Sys.time()

paste0(round(as.numeric(difftime(time1 = svm_cv_b, time2 = svm_cv_a, units = "secs")), 3), " Elapsed ti
```

```
[1] "16.637 Elapsed time in Seconds"
```

```
row.names_2 <-  c('CV Logistic Regression',
               'CV SVM')
acc_2 <- c(
         round(logCvMat$overall['Accuracy'], 3),
         round(svmConfMat_2$overall['Accuracy'], 3))

sens_2 <- c(
         round(logCvMat$byClass['Sensitivity'], 3),
         round(svmConfMat_2$byClass['Sensitivity'], 3))

spec_2 <- c(
         round(logCvMat$byClass['Specificity'], 3),
         round(svmConfMat_2$byClass['Specificity'], 3))

# auc <- c(
#          round(auc(logRegConfMat),3),
#          round(auc(svmConfMat), 3))
kappa_2 <- c(
         round(logCvMat$overall['Kappa'], 3),
         round(svmConfMat_2$overall['Kappa'], 3))
F1_2 <- c(
         round(logCvMat$byClass['F1'], 3),
         round(svmConfMat_2$byClass['F1'], 3))

sum_df_cv <- tibble(Model = row.names_2,
                 Accuracy = acc_2,
                 Sensitivity = sens_2,
                 Specificity = spec_2,
                 Kappa = kappa_2,
                 # Auc = auc,
                 F1 = F1_2)
sum_df_cv
```

```
# A tibble: 2 x 6
  Model                  Accuracy Sensitivity Specificity Kappa    F1
  <chr>                     <dbl>       <dbl>       <dbl> <dbl> <dbl>
```

```
1 CV Logistic Regression    0.847       0.875       0.815 0.692 0.862
2 CV SVM                    0.831       0.812       0.852 0.661 0.839
```

Bootstrap - Logistic regression

```r
set.seed(43)
log_boot_a = Sys.time()
# for this to work add names to all levels (numbers not allowed)
feature.names=names(heart.data)

for (f in feature.names) {
  if (class(heart.data[[f]])=="factor") {
    levels <- unique(c(heart.data[[f]]))
    heart.data[[f]] <- factor(heart.data[[f]],
                   labels=make.names(levels))
  }
}

inTrainRows <- createDataPartition(heart.data$num,p=0.8,list=FALSE)
trainData3 <- heart.data[inTrainRows,]
testData3 <-  heart.data[-inTrainRows,]

# Kfold control
ctr_boot <- trainControl(method = "boot",
                           number = 200,
                           classProbs = TRUE,
                           summaryFunction = twoClassSummary)

# k-fold training
log_boot <- train(num ~ ., data = trainData3,
                 method = "glm",
                 family = "binomial",
                 trControl = ctr_boot,
                 preProcess = c("center", "scale"),
                 metric = "ROC")

log_boot_pred <- predict(log_boot, testData3)
log_boot_Pred_prob <- predict(log_boot, testData3, type='prob')[2]
(logbootMat <- caret::confusionMatrix(log_boot_pred, testData3[,"num"]))
```

```
Confusion Matrix and Statistics

          Reference
Prediction X1 X2
        X1 28  5
        X2  4 22

               Accuracy : 0.8475
                 95% CI : (0.7301, 0.9278)
    No Information Rate : 0.5424
    P-Value [Acc > NIR] : 7.195e-07

                  Kappa : 0.6918

 Mcnemar's Test P-Value : 1
```

```
            Sensitivity : 0.8750
            Specificity : 0.8148
         Pos Pred Value : 0.8485
         Neg Pred Value : 0.8462
             Prevalence : 0.5424
         Detection Rate : 0.4746
   Detection Prevalence : 0.5593
      Balanced Accuracy : 0.8449

       'Positive' Class : X1
```

```
log_boot_b = Sys.time()
#ROC Curve
# AUC$svm <- roc(as.numeric(testData2$num),as.numeric(as.matrix((log_cv_Pred_prob))))$auc
# Accuracy$svm <- svmConfMat$overall['Accuracy']
paste0(round(as.numeric(difftime(time1 = log_boot_b, time2 = log_boot_a, units = "secs")), 3), " Elapsed
```

```
[1] "7.291 Elapsed time in Seconds"
```

Bootstrap - SVM

```r
set.seed(43)
svm_boot_a = Sys.time()
# for this to work add names to all levels (numbers not allowed)
feature.names=names(heart.data)

for (f in feature.names) {
  if (class(heart.data[[f]])=="factor") {
    levels <- unique(c(heart.data[[f]]))
    heart.data[[f]] <- factor(heart.data[[f]],
                   labels=make.names(levels))
  }
}

inTrainRows <- createDataPartition(heart.data$num,p=0.8,list=FALSE)
trainData4 <- heart.data[inTrainRows,]
testData4 <-  heart.data[-inTrainRows,]

fitControl_4 <- trainControl(method = "boot",
                           ## Estimate class probabilities
                           classProbs = TRUE,
                           repeats = 200,
                           summaryFunction = twoClassSummary)

svmModel_4 <- train(num ~ ., data = trainData2,
                 method = "svmRadial",
                 trControl = fitControl_4,
                 preProcess = c("center", "scale"),
                 metric = "ROC")

svmPrediction_4 <- predict(svmModel_4, testData4)
svmPredictionprob_4 <- predict(svmModel_4, testData4, type='prob')[2]
(svmConfMat_4 <- caret::confusionMatrix(svmPrediction_4, testData4[,"num"]))
```

```
Confusion Matrix and Statistics

          Reference
Prediction X1 X2
        X1 26  7
        X2  6 20

               Accuracy : 0.7797
                 95% CI : (0.6527, 0.8771)
    No Information Rate : 0.5424
    P-Value [Acc > NIR] : 0.0001366

                  Kappa : 0.5548

 Mcnemar's Test P-Value : 1.0000000

            Sensitivity : 0.8125
            Specificity : 0.7407
```

```
         Pos Pred Value : 0.7879
         Neg Pred Value : 0.7692
             Prevalence : 0.5424
         Detection Rate : 0.4407
   Detection Prevalence : 0.5593
      Balanced Accuracy : 0.7766

       'Positive' Class : X1
```

```
svm_boot_b = Sys.time()
#ROC Curve
# AUC$svm <- roc(as.numeric(testData2$num),as.numeric(as.matrix((svmPredictionprob))))$auc
# Accuracy$svm_boot <- svmConfMat$overall['Accuracy']
paste0(round(as.numeric(difftime(time1 = svm_boot_b, time2 = svm_boot_a, units = "secs")), 3), " Elapse
```

```
[1] "7.064 Elapsed time in Seconds"
```

```
row.names_4 <-  c('Boot Logistic Regression',
               'Boot SVM')
acc_4 <- c(
         round(logbootMat$overall['Accuracy'], 3),
         round(svmConfMat_4$overall['Accuracy'], 3))

sens_4 <- c(
         round(logbootMat$byClass['Sensitivity'], 3),
         round(svmConfMat_4$byClass['Sensitivity'], 3))

spec_4 <- c(
         round(logbootMat$byClass['Specificity'], 3),
         round(svmConfMat_4$byClass['Specificity'], 3))

# auc <- c(
#           round(auc(logRegConfMat),3),
#           round(auc(svmConfMat), 3))
kappa_4 <- c(
         round(logbootMat$overall['Kappa'], 3),
         round(svmConfMat_4$overall['Kappa'], 3))
F1_4 <- c(
         round(logbootMat$byClass['F1'], 3),
         round(svmConfMat_4$byClass['F1'], 3))

sum_df_boot <- tibble(Model = row.names_4,
                 Accuracy = acc_4,
                 Sensitivity = sens_4,
                 Specificity = spec_4,
                 Kappa = kappa_4,
                 # Auc = auc,
                 F1 = F1_4)
sum_df_boot
```

```
# A tibble: 2 x 6
  Model                    Accuracy Sensitivity Specificity Kappa    F1
  <chr>                         <dbl>       <dbl>       <dbl> <dbl> <dbl>
1 Boot Logistic Regression    0.847       0.875       0.815 0.692 0.862
2 Boot SVM                    0.78        0.812       0.741 0.555 0.8
```

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
inTrainRows <- createDataPartition(heart.data$num,p=0.8,list=FALSE)
trainData_f <- heart.data[inTrainRows,]
testData_f <-  heart.data[-inTrainRows,]
nrow(trainData_f)/(nrow(testData_f)+nrow(trainData_f)) #checking whether really 80% -> OK
```

```
[1] 0.8013468
```

```r
set.seed(43)
rf_a = Sys.time()
library(randomForest)
RFModel <- randomForest(num ~ .,
                    data=trainData,
                    importance=TRUE,
                    ntree=2000)
#varImpPlot(RFModel)
RFPrediction <- predict(RFModel, testData)
RFPredictionprob = predict(RFModel,testData,type="prob")[, 2]

(RFConfMat <- caret::confusionMatrix(RFPrediction, testData[,"num"]))
```

```
Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0 27  6
         1  5 21

               Accuracy : 0.8136
                 95% CI : (0.6909, 0.9031)
    No Information Rate : 0.5424
    P-Value [Acc > NIR] : 1.224e-05

                  Kappa : 0.6233

 Mcnemar's Test P-Value : 1

            Sensitivity : 0.8438
            Specificity : 0.7778
         Pos Pred Value : 0.8182
         Neg Pred Value : 0.8077
             Prevalence : 0.5424
         Detection Rate : 0.4576
   Detection Prevalence : 0.5593
      Balanced Accuracy : 0.8108
```

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

```
rf_b = Sys.time()
paste0(round(as.numeric(difftime(time1 = rf_b, time2 = rf_a, units = "secs")), 3), " Elapsed time in Sec
```

```
[1] "1.357 Elapsed time in Seconds"
```

```
row.names <-  c('Basic Logistic Regression',
               'Basic SVM',
               'CV Logistic Regression',
               'CV SVM',
               'Boot Logistic Regression',
               'Boot SVM',
               'Random Forest')
acc <- c(round(logRegConfMat$overall['Accuracy'], 3),
         round(svmConfMat$overall['Accuracy'], 3),
         round(logCvMat$overall['Accuracy'], 3),
         round(svmConfMat_2$overall['Accuracy'], 3),
         round(logbootMat$overall['Accuracy'], 3),
         round(svmConfMat_4$overall['Accuracy'], 3),
         round(RFConfMat$overall['Accuracy'], 3))

sens <- c(round(logRegConfMat$byClass['Sensitivity'], 3),
         round(svmConfMat$byClass['Sensitivity'], 3),
         round(logCvMat$byClass['Sensitivity'], 3),
         round(svmConfMat_2$byClass['Sensitivity'], 3),
         round(logbootMat$byClass['Sensitivity'], 3),
         round(svmConfMat_4$byClass['Sensitivity'], 3),
         round(RFConfMat$byClass['Sensitivity'], 3))

spec <- c(round(logRegConfMat$byClass['Specificity'], 3),
         round(svmConfMat$byClass['Specificity'], 3),
         round(logCvMat$byClass['Specificity'], 3),
         round(svmConfMat_2$byClass['Specificity'], 3),
         round(logbootMat$byClass['Specificity'], 3),
         round(svmConfMat_4$byClass['Specificity'], 3),
         round(RFConfMat$byClass['Specificity'], 3))


kappa <- c(round(logRegConfMat$overall['Kappa'], 3),
         round(svmConfMat$overall['Kappa'], 3),
         round(logCvMat$overall['Kappa'], 3),
         round(svmConfMat_2$overall['Kappa'], 3),
         round(logbootMat$overall['Kappa'], 3),
         round(svmConfMat_4$overall['Kappa'], 3),
         round(RFConfMat$overall['Kappa'], 3))

F1 <- c(round(logRegConfMat$byClass['F1'], 3),
         round(svmConfMat$byClass['F1'], 3),
         round(logCvMat$byClass['F1'], 3),
         round(svmConfMat_2$byClass['F1'], 3),
         round(logbootMat$byClass['F1'], 3),
```

```r
         round(svmConfMat_4$byClass['F1'], 3),
         round(RFConfMat$byClass['F1'], 3))
time <- c(round(as.numeric(difftime(time1 = log_base_b, time2 = log_base_a, units = "secs")), 3),
          round(as.numeric(difftime(time1 = svm_base_b, time2 = svm_base_a, units = "secs")), 3),
          round(as.numeric(difftime(time1 = log_cv_b, time2 = log_cv_a, units = "secs")), 3),
          round(as.numeric(difftime(time1 = svm_cv_b, time2 = svm_cv_a, units = "secs")), 3),
          round(as.numeric(difftime(time1 = log_boot_b, time2 = log_boot_a, units = "secs")), 3),
          round(as.numeric(difftime(time1 = svm_boot_b, time2 = svm_boot_a, units = "secs")), 3),
          round(as.numeric(difftime(time1 = rf_b, time2 = rf_a, units = "secs")), 3))

sum_df <- tibble(Model = row.names,
                 Accuracy = acc,
                 Sensitivity = sens,
                 Specificity = spec,
                 Kappa = kappa,
                 F1 = F1,
                 Time = time)
sum_df
```

```
# A tibble: 7 x 7
  Model                      Accuracy Sensitivity Specificity Kappa    F1  Time
  <chr>                         <dbl>       <dbl>       <dbl> <dbl> <dbl> <dbl>
1 Basic Logistic Regression     0.847       0.875       0.815 0.692 0.862  2.86
2 Basic SVM                     0.78        0.812       0.741 0.555 0.8    2.97
3 CV Logistic Regression        0.847       0.875       0.815 0.692 0.862  2.99
4 CV SVM                        0.831       0.812       0.852 0.661 0.839 16.6
5 Boot Logistic Regression      0.847       0.875       0.815 0.692 0.862  7.29
6 Boot SVM                      0.78        0.812       0.741 0.555 0.8    7.06
7 Random Forest                 0.814       0.844       0.778 0.623 0.831  1.36
```

Part C - Summary

14 predictor variables from the UCI heart disease dataset were used to predict the diagnosis of heart disease (angiographic disease status). The performances of 7 different machine learning algorithms using logistic regression and support vector machine as basic model, boosted trees and random forest - are compared .

A comparison of the area under the Kappa and the accuracy of the model predictions shows that logistic regression performs best (accuracy of 0.847, F1 0.862). Tree-based methods performed slighly worse.

Nevertheless, the Random forest model shows a significant training elapsed time compared to other boosted trees (~ 2 seconds compared to cross validation SVM more than 10 seconds and boot logistic regression of 7 seconds). However, Random forest took less time to train compared to the Logistic Regression Base Model which trained the model in almost double the time ~ 3 seconds.

For this dataset, I assumed the Ockham's Razor where I assumed that the data has enough variance and baise balance to explain the model. In other words, the base logistic model can be used to predict the heart disease (angiographic disease status). After conducting the necessary analysis, it showed that this assumption may be valid and could implemented from the beginning.