590-01 Final Project

Roderick Whang

4/6/2021

```
rm(list = ls())
setwd("C:\\590_final")
library(tidyverse)
library(ggplot2)
library(lubridate)
library(patchwork)
library(gridExtra)
library(psych)
library(corrplot)
library(ggfortify)
```

```
## Warning: package 'ggfortify' was built under R version 4.0.4
```

```
library(factoextra)
```

```
## Warning: package 'factoextra' was built under R version 4.0.4
```

```
library(class) #knn
library(gmodels) # CrossTable()
```

```
## Warning: package 'gmodels' was built under R version 4.0.4
```

```
library(caret) # creatFolds()
library(caTools) #sample.split()
```

```
## Warning: package 'caTools' was built under R version 4.0.4
```

```
library(ROCR) # prediction(), performance()
```

```
## Warning: package 'ROCR' was built under R version 4.0.4
```

```
library(randomForest) # Random Forest
library(caret)
library(e1071) # SVM
set.seed(2021)

df1 <- read.csv("S1.csv")
df2 <- read.csv("S2.csv")
df3 <- read.csv("S3.csv")
df4 <- read.csv("S4.csv")
df5 <- read.csv("S5.csv")
df6 <- read.csv("S6.csv")
df7 <- read.csv("S7.csv")
df8 <- read.csv("S8.csv")
df9 <- read.csv("S9.csv")
df10 <- read.csv("S10.csv")
```

```r
df11 <- read.csv("S11.csv")
df12 <- read.csv("S12.csv")
df13 <- read.csv("S13.csv")
df14 <- read.csv("S14.csv")
df15 <- read.csv("S15.csv")

df <- rbind(df1, df2, df3, df4, df5, df6, df7, df8, df9, df10, df11, df12, df13, df14, df15)
df$activity <- as.factor(df$activity)
df <- subset(df, select = -c(X) )
df <- subset(df, df$activity == 1 | df$activity == 2 | df$activity == 3 |
               df$activity ==4 | df$activity == 5| df$activity == 6 | df$activity == 7 |
               df$activity ==8)

# df$Activity
# low = 1

# medium = 3
df$activity[df$activity== 3 | df$activity== 5 | df$activity == 6 | df$activity == 8] <- 3

# high = 2
df$activity[df$activity==2 | df$activity== 4 | df$activity == 7] <- 2


df$activity <- factor(df$activity)

a2 <- which(df$activity ==2)
a3 <- which(df$activity ==3)

a2_s <- sample(a2, 6000, replace = FALSE)
a3_s <- sample(a3, 26000, replace = FALSE)

df <- df[-c(a2_s, a3_s),]
```

```r
head(df)
```

```
##    activity      f1.mean    f2.std    f3.max     f4.min f5.max_position
## 45        1 -0.038429169 0.3478793 0.4465136 -1.0299322     0.070357143
## 46        1 -0.011527297 0.3137633 0.4326159 -0.7738035     0.259821429
## 47        1  0.005274472 0.2966080 0.4326159 -0.7738035     0.009821429
## 48        1 -0.015503497 0.2861047 0.3738340 -0.7559939     0.086071429
## 49        1  0.004943980 0.2922504 0.4119239 -0.8208496     0.980535714
## 50        1 -0.005607502 0.2965154 0.4475430 -0.8208496     0.970714286
##    f6.min_position    f7.hr f8.skewness f9.kurtosis
## 45      0.01375000 46.11024  -0.7152854 -0.51051998
## 46      0.37500000 46.50810  -0.7118847 -0.55704806
## 47      0.12500000 47.03796  -0.8300902 -0.19700263
## 48      0.02946429 46.88382  -0.7594123 -0.36029498
## 49      0.92196429 46.32763  -0.8617181 -0.01984442
## 50      0.67196429 45.61259  -0.8270814 -0.03742761
```

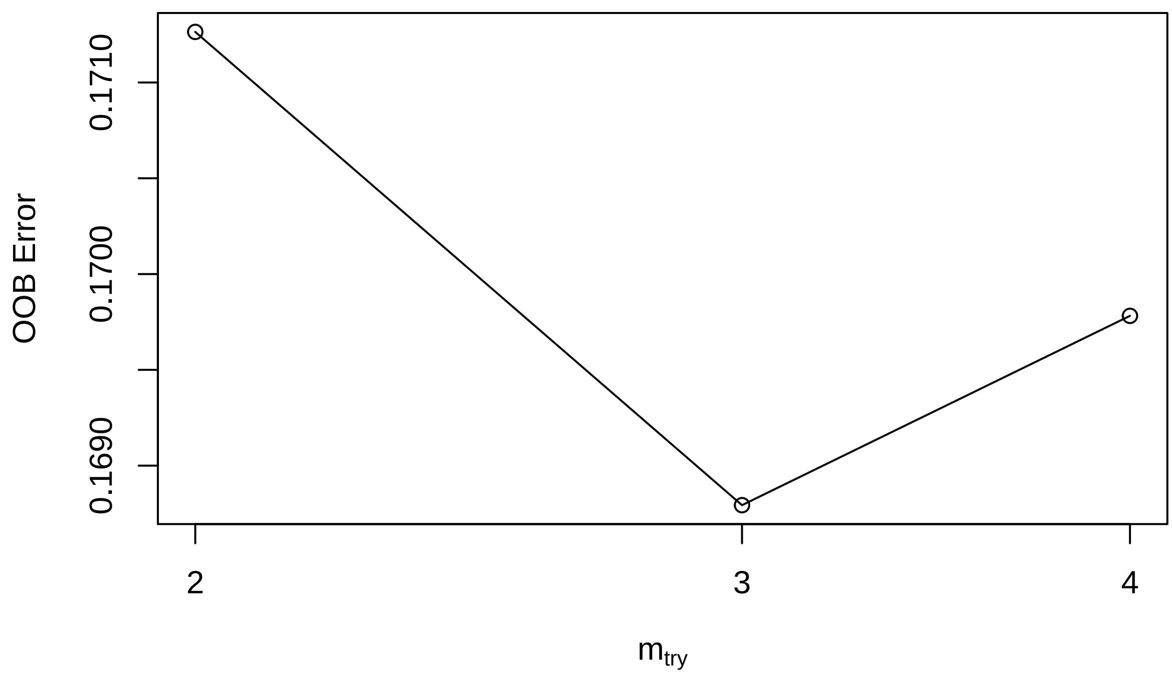
```r
set.seed(2021)

sample <- sample.split(df$activity, SplitRatio = .8) # dataset to split it into 80:20
```

```
df_train <- df[sample==TRUE, ]
df_test <- df[sample==FALSE, ]

X_train <- subset(df_train, select = -c(activity) ) # independent variables
y_train <- df_train[,1] # tartget variables

X_test <- subset(df_test, select = -c(activity) ) # independent variables
y_test <- df_test[,1] # tartget variables
```

RF

```
bestmtry <- tuneRF(X_train, y_train, stepFactor=1.5, improve=1e-5, ntree=700)
```

```
## mtry = 3  OOB error = 16.88%
## Searching left ...
## mtry = 2     OOB error = 17.13%
## -0.01463415 1e-05
## Searching right ...
## mtry = 4     OOB error = 16.98%
## -0.005853659 1e-05
```




```
print(bestmtry)
```

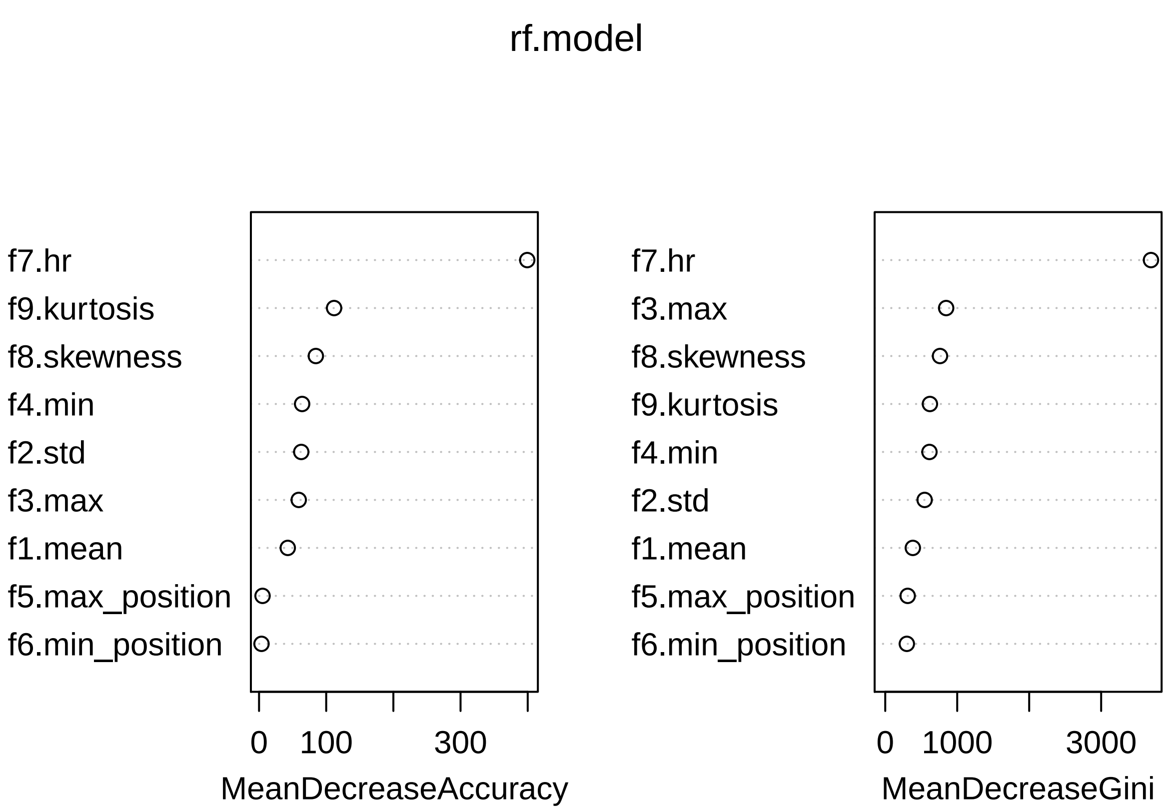
```
##       mtry  OOBError
## 2.OOB    2 0.1712639
## 3.OOB    3 0.1687937
```

```
## 4.OOB    4 0.1697818
```

```
rf.model <- randomForest(formula = activity ~ ., data = df_train, ntree=700, mtry=3, importance = TRUE,
rf.model
```

```
##
## Call:
##  randomForest(formula = activity ~ ., data = df_train, ntree = 700,      mtry = 3, importance = TRUE
##                Type of random forest: classification
##                      Number of trees: 700
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 16.95%
## Confusion matrix:
##      1    2    3 class.error
## 1 3331   94  230  0.08864569
## 2    7 3684  636  0.14860180
## 3  243  848 3072  0.26207062
```

```
varImpPlot(rf.model)
```



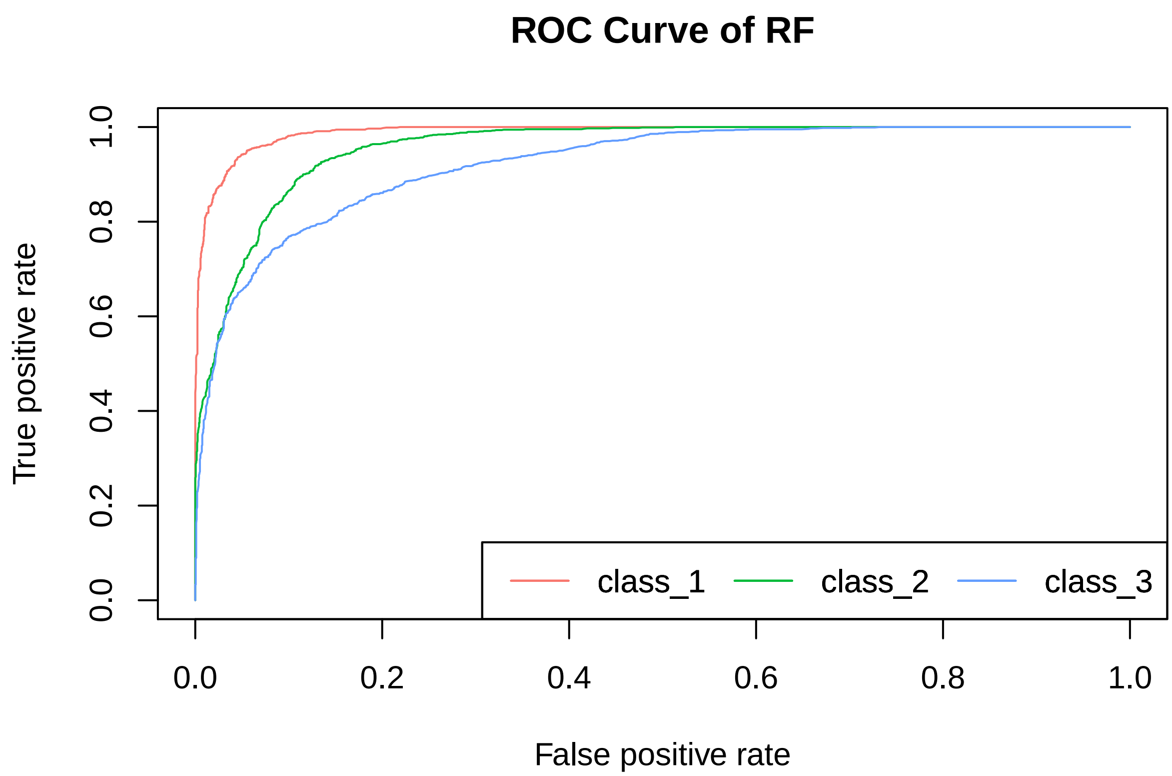
The Mean Decrease Accuracy plot expresses how much accuracy the model losses by excluding each variable. The more the accuracy suffers, the more important the variable is for the successful classification. The variables are presented from descending importance. The mean decrease in Gini coefficient is a measure of how each variable contributes to the homogeneity of the nodes and leaves in the resulting random forest. The higher the value of mean decrease accuracy or mean decrease Gini score, the higher the importance of the variable in the model.

```r
prediction_for_table <- predict(rf.model,X_test)
#table(observed=y_test,predicted=prediction_for_table)
confusionMatrix(prediction_for_table, y_test)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   1   2   3
##          1 825   1  69
##          2  26 943 178
##          3  63 138 794
##
## Overall Statistics
##
##                Accuracy : 0.8436
##                  95% CI : (0.8302, 0.8563)
##     No Information Rate : 0.3563
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7646
##
##  Mcnemar's Test P-Value : 2.874e-06
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3
## Sensitivity            0.9026   0.8715   0.7627
## Specificity            0.9670   0.8957   0.8993
## Pos Pred Value         0.9218   0.8221   0.7980
## Neg Pred Value         0.9585   0.9265   0.8790
## Prevalence             0.3010   0.3563   0.3428
## Detection Rate         0.2716   0.3105   0.2614
## Detection Prevalence   0.2947   0.3777   0.3276
## Balanced Accuracy      0.9348   0.8836   0.8310
```

```r
pred_prob.rf <- predict(rf.model, X_test, decision.values = TRUE, type="prob")
colours <- c("#F8766D","#00BA38","#619CFF")
# Specify the different classes
classes <- levels(df$activity)
# For each class
for (i in 1:3)
{
 # Define which observations belong to class[i]
 true_values <- ifelse(y_test==classes[i],1,0)
 # Assess the performance of classifier for class[i]
 pred <- prediction(pred_prob.rf[,i],true_values)

 perf <- performance(pred, "tpr", "fpr")
 if (i==1)
 {
     plot(perf,main="ROC Curve of RF",col=colours[i])
 }
 else
 {
```

```
    plot(perf,main="ROC Curve of RF",col=colours[i],add=TRUE)
 }
 legend("bottomright",  c("class_1","class_2","class_3"), col = colours, lty= 1, horiz=TRUE)
 # Calculate the AUC and print it to screen
 auc.perf <- performance(pred, measure = "auc")
 print(paste("AUC of class_",i,":",auc.perf@y.values))
}
```




```
## [1] "AUC of class_ 1 : 0.989296658149619"
## [1] "AUC of class_ 2 : 0.956864478492514"
## [1] "AUC of class_ 3 : 0.921564310176549"
```

SVM

```
#set.seed(1)
#X <- sample(dim(X_train)[1], 3000, replace=FALSE)
#tune.out <- tune(svm, activity ~., data=df_train[X,],
#                 kernel='radial',
#                 ranges = list(cost=c(0.1,1,10,100,1000),
#                 gamma=c(0.5, 1,2,3,4)))

#summary(tune.out)
```

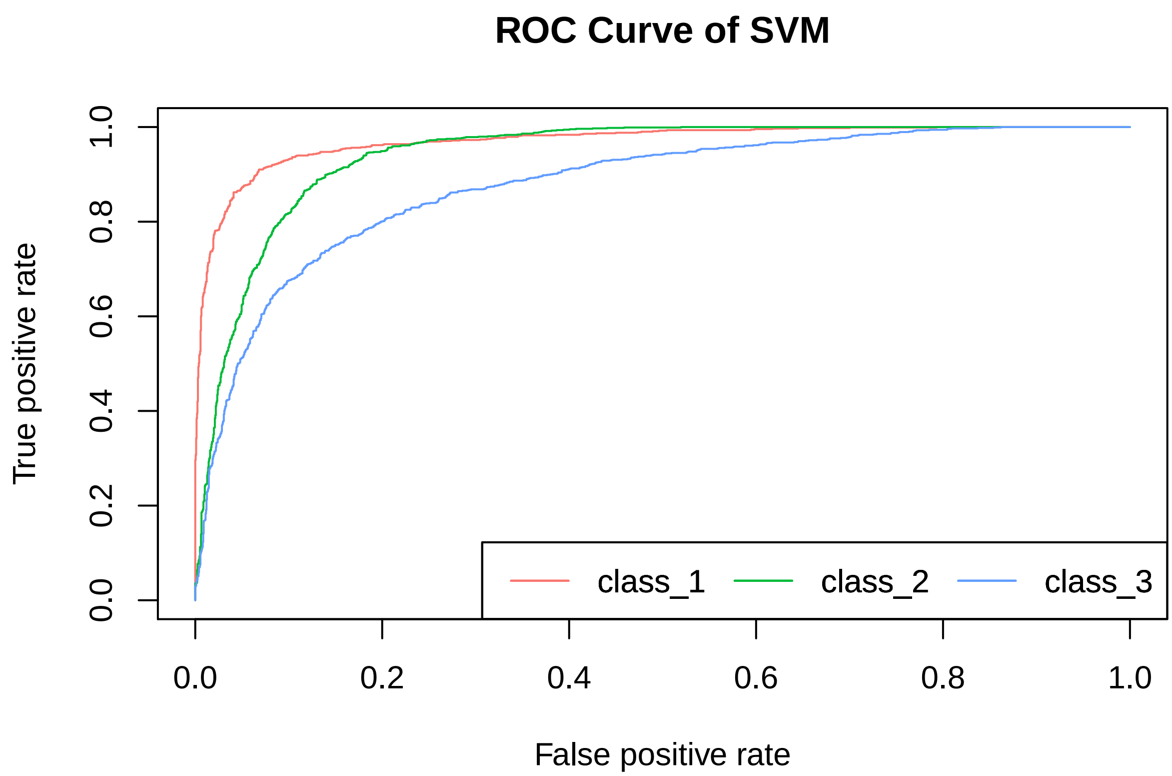
```
svm.opt <- svm(activity ~., data=df_train, kernel='radial', type =  'C-classification',
               gamma=0.07, cost=10
               , decision.values=T, probability = TRUE)
```

```r
pred <- predict(svm.opt, X_test, decision.values = TRUE, probability = TRUE)

confusionMatrix(pred, y_test)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   1   2   3
##          1 796   2  93
##          2  27 918 200
##          3  91 162 748
##
## Overall Statistics
##
##                Accuracy : 0.8107
##                  95% CI : (0.7963, 0.8245)
##     No Information Rate : 0.3563
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.715
##
##  Mcnemar's Test P-Value : 1.178e-05
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3
## Sensitivity            0.8709   0.8484   0.7185
## Specificity            0.9553   0.8839   0.8732
## Pos Pred Value         0.8934   0.8017   0.7473
## Neg Pred Value         0.9450   0.9133   0.8561
## Prevalence             0.3010   0.3563   0.3428
## Detection Rate         0.2621   0.3023   0.2463
## Detection Prevalence   0.2934   0.3770   0.3296
## Balanced Accuracy      0.9131   0.8662   0.7959
```

```r
pred_prob.svm <- attr(pred, "probabilities")
colours <- c("#F8766D","#00BA38","#619CFF")
# Specify the different classes
classes <- levels(df$activity)
# For each class
for (i in 1:3)
{
 # Define which observations belong to class[i]
 true_values <- ifelse(y_test==classes[i],1,0)
 # Assess the performance of classifier for class[i]
 pred <- prediction(pred_prob.svm[,i],true_values)

 perf <- performance(pred, "tpr", "fpr")
 if (i==1)
 {
     plot(perf,main="ROC Curve of SVM",col=colours[i])
 }
 else
 {
```

```
    plot(perf,main="ROC Curve of SVM",col=colours[i],add=TRUE)
 }
 legend("bottomright",  c("class_1","class_2","class_3"), col = colours, lty= 1, horiz=TRUE)
 # Calculate the AUC and print it to screen
 auc.perf <- performance(pred, measure = "auc")
 print(paste("AUC of class_",i,":",auc.perf@y.values))
}
```




```
## [1] "AUC of class_ 1 : 0.969947258895237"
## [1] "AUC of class_ 2 : 0.94195082517456"
## [1] "AUC of class_ 3 : 0.877091839779471"
```