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)

df <- read.csv("S1.csv")
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
```

```r
# 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)
```

```r
head(df)
```

```
##    activity      f1.mean    f2.std    f3.max     f4.min f5.max_position
## 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
## 51        1  0.005737323 0.2996692 0.4475430 -0.8208496     0.720714286
##    f6.min_position    f7.hr f8.skewness f9.kurtosis
## 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
## 51      0.42196429 46.20430  -0.9563866  0.06290094
```

```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 <- df_train[,2:10] # independent variables
y_train <- df_train[,1] # tartget variables

X_test <- df_test[,2:10] # independent variables
y_test <- df_test[,1] # tartget variables
```

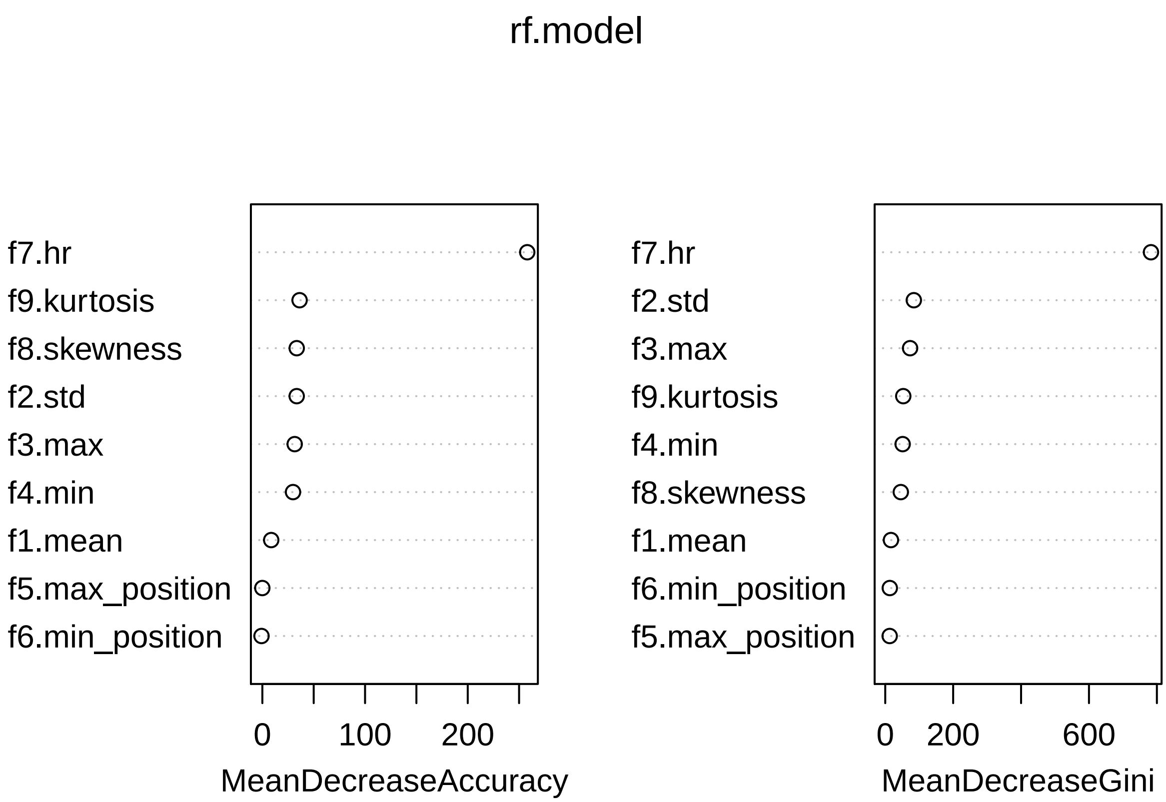
RF

```r
rf.model <- randomForest(formula = activity ~ ., data = df_train, ntree=500, mtry=3, importance = TRUE,
rf.model
```

```
##
## Call:
##  randomForest(formula = activity ~ ., data = df_train, ntree = 500,      mtry = 3, importance = TRUE
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 3.45%
## Confusion matrix:
##     1   2    3 class.error
```

```
## 1 132   2   31  0.20000000
## 2   1 550   29  0.05172414
## 3   6  20 1812  0.01414581
```

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



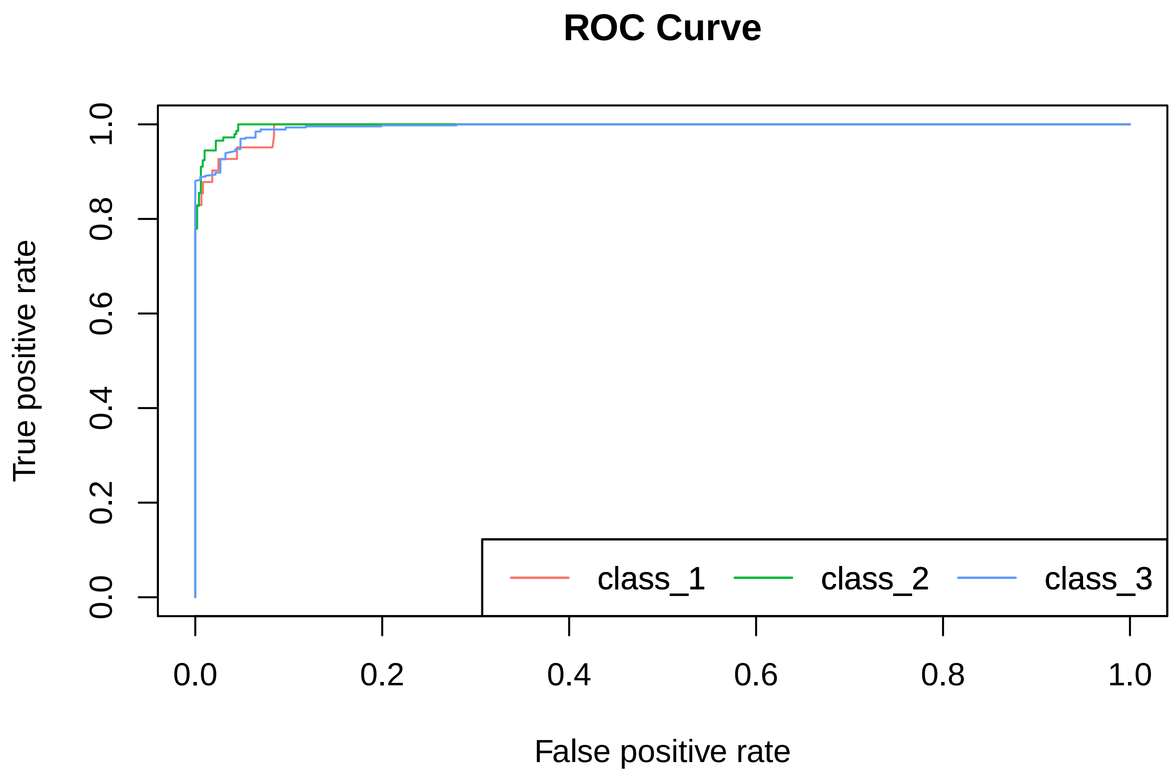
The Mean Decrease Accuracy plot expresses how much accuracy the model losses by excluding each variable. The more the accuracy suffers, the more important the variable is for the successful classification. The variables are presented from descending importance. The mean decrease in Gini coefficient is 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  32   0   0
##          2   2 131   2
##          3   7  14 458
##
## Overall Statistics
##
##                Accuracy : 0.9613
```

```
##                   95% CI : (0.9434, 0.9748)
##      No Information Rate : 0.7121
##      P-Value [Acc > NIR] : < 2.2e-16
##
##                    Kappa : 0.9083
##
##   Mcnemar's Test P-Value : 0.0004398
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3
## Sensitivity           0.78049   0.9034   0.9957
## Specificity           1.00000   0.9920   0.8871
## Pos Pred Value        1.00000   0.9704   0.9562
## Neg Pred Value        0.98534   0.9726   0.9880
## Prevalence            0.06347   0.2245   0.7121
## Detection Rate        0.04954   0.2028   0.7090
## Detection Prevalence  0.04954   0.2090   0.7415
## Balanced Accuracy     0.89024   0.9477   0.9414
```

```
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",col=colours[i])
 }
 else
 {
     plot(perf,main="ROC Curve",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 : 0.993408586978432"
## [1] "AUC of class_ 2 : 0.99726065111157"
## [1] "AUC of class_ 3 : 0.994243805516596"
```

SVM

```r
set.seed(1)
tune.out <- tune(svm, activity ~., data=df_train,
                 kernel='radial',
                 ranges = list(cost=c(0.1,1,10,100,1000),
                 gamma=c(0.5, 1,2,3,4)))

summary(tune.out)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost gamma
##    10   0.5
##
## - best performance: 0.05071234
##
## - Detailed performance results:
```

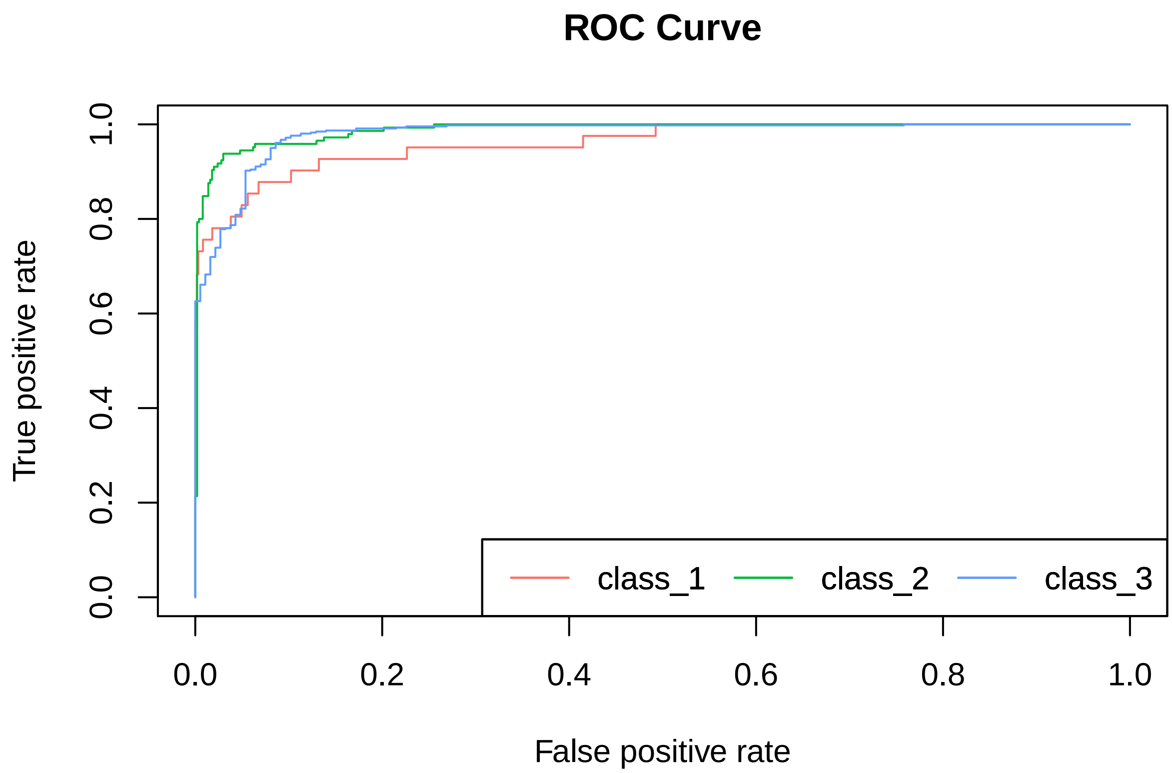
```
##     cost gamma      error  dispersion
## 1  1e-01   0.5 0.11577026 0.014769147
## 2  1e+00   0.5 0.05342851 0.008738338
## 3  1e+01   0.5 0.05071234 0.005258890
## 4  1e+02   0.5 0.05729550 0.007464743
## 5  1e+03   0.5 0.05729550 0.007464743
## 6  1e-01   1.0 0.20095029 0.019753363
## 7  1e+00   1.0 0.07627129 0.011869722
## 8  1e+01   1.0 0.07278441 0.009104946
## 9  1e+02   1.0 0.07317201 0.008273443
## 10 1e+03   1.0 0.07317201 0.008273443
## 11 1e-01   2.0 0.28652240 0.027862045
## 12 1e+00   2.0 0.13319266 0.016973319
## 13 1e+01   2.0 0.12195983 0.018166372
## 14 1e+02   2.0 0.12195983 0.018166372
## 15 1e+03   2.0 0.12195983 0.018166372
## 16 1e-01   3.0 0.28845739 0.027146439
## 17 1e+00   3.0 0.18545838 0.013183236
## 18 1e+01   3.0 0.16338631 0.012637302
## 19 1e+02   3.0 0.16338631 0.012637302
## 20 1e+03   3.0 0.16338631 0.012637302
## 21 1e-01   4.0 0.28845739 0.027146439
## 22 1e+00   4.0 0.22495436 0.019602298
## 23 1e+01   4.0 0.20675526 0.018201529
## 24 1e+02   4.0 0.20675526 0.018201529
## 25 1e+03   4.0 0.20675526 0.018201529
```

```r
svm.opt <- svm(activity ~., data=df_train, kernel='radial',
               gamma=0.5, cost=10, decision.values=T, probability = TRUE)
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  30   1   1
##          2   2 130   7
##          3   9  14 452
##
## Overall Statistics
##
##                Accuracy : 0.9474
##                  95% CI : (0.9272, 0.9633)
##     No Information Rate : 0.7121
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.8762
##
##  Mcnemar's Test P-Value : 0.02842
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3
```

```
## Sensitivity           0.73171   0.8966   0.9826
## Specificity           0.99669   0.9820   0.8763
## Pos Pred Value        0.93750   0.9353   0.9516
## Neg Pred Value        0.98208   0.9704   0.9532
## Prevalence            0.06347   0.2245   0.7121
## Detection Rate        0.04644   0.2012   0.6997
## Detection Prevalence  0.04954   0.2152   0.7353
## Balanced Accuracy     0.86420   0.9393   0.9295
```

```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",col=colours[i])
 }
 else
 {
     plot(perf,main="ROC Curve",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 : 0.960491836323322"
## [1] "AUC of class_ 2 : 0.988175373391149"
## [1] "AUC of class_ 3 : 0.980025712949974"
```