**Random Forest**

Random forest is a model that generates classification output through growing numerous random decision trees and using each tree’s classification output to “vote” for the best output.

data.forest <- randomForest(

weight\_category ~ .,

data = data,

subset = train,

ntree = 5000,

importance = T,

localImp = T

)

data.forest

##

## Call:

## randomForest(formula = weight\_category ~ ., data = data, ntree = 5000, importance = T, localImp = T, subset = train)

## Type of random forest: classification

## Number of trees: 5000

## No. of variables tried at each split: 4

## OOB estimate of error rate: 5.5%

In the random forest model, assuming the number of trees is always sufficiently big, the most effective hyperparameter is the number of variables that the algorithm tried at each split (mtry attribute in the randomForest function). Like what we did with selecting the best decision tree, the first step is to set up a baseline forest, where mtry = ncol(data)-1.

data.forest <- randomForest(

weight\_category ~ .,

data = data,

subset = train,

ntree = 5000,

mtry = ncol(data) - 1,

importance = T,

localImp = T

)

data.forest

##

## Call:

##randomForest(formula = weight\_category ~ ., data = data, ntree = 5000, mtry = ncol(data) - 1, importance = T, localImp = T, subset = train)

## Type of random forest: classification

## Number of trees: 5000

## No. of variables tried at each split: 16

## OOB estimate of error rate: 4.83%

data.forest.pred <- predict(data.forest, test, typr = "class")

mean(data.forest.pred == test$weight\_category)

## [1] 0.9422348

The baseline forest with mtry of 16 already demonstrates better performance than a single decision tree, having 94.22% accuracy compared to 84.47% on the testing data.

mtry.list <- c()

data.forest.final <- NULL

max <- 0

for (i in 1:ncol(data) - 1) {

data.forest.dummy <- randomForest(

weight\_category ~ .,

data = data,

subset = train,

ntree = 5000,

mtry = i,

importance = T,

localImp = T

)

val <- predict(data.forest.dummy, test, typr = "class")

score <- mean(val == test$weight\_category)

if (score >= max) {

data.forest.final <- data.forest.dummy

max <- score

}

mtry.list[i] <- score

}

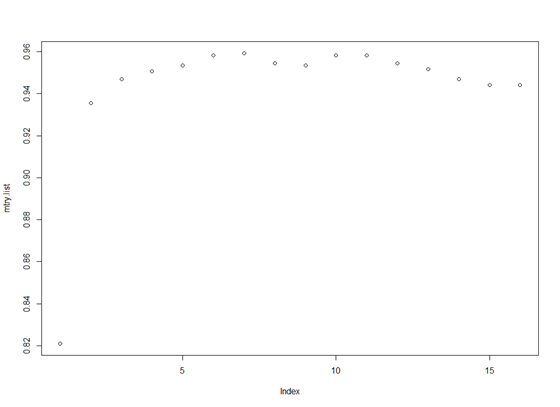
mtry.list

[1]0.8210227 0.9356061 0.9469697 0.9507576 0.9535985 0.9583333 0.9592803

[8]0.9545455 0.9535985 0.9583333 0.9583333 0.9545455 0.9517045 0.9469697

[15]0.9441288 0.9441288

plot(mtry.list)



Using a for loop we tested out all feasible mtry values and found a forest with the highest accuracy. (below is the best forest)

data.forest.final

##

## Call:

## randomForest(formula = weight\_category ~ ., data = data, ntree = 5000, mtry = i, importance = T, localImp = T, subset = train)

## Type of random forest: classification

## Number of trees: 5000

## No. of variables tried at each split: 7

## OOB estimate of error rate: 4.74%

data.forest.final.pred <-

predict(data.forest.final, test, typr = "class")

mean(data.forest.final.pred == test$weight\_category)

## [1] 0.9592803

The final forest with a mtry of 7 and a size of 5000, generated an accuracy of 95.93% (1.8% improvements compared to the baseline forest).