RandomForest

Scott McCoy - U80152879

2/24/2021

Random Forest with K-Fold CV

Libraries:

```
set.seed(430)
library(data.table)
library(ggplot2)
library(ggthemes)
library(scales)
library(randomForest)
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
theme_set(theme_bw())
```

Train-Test Split

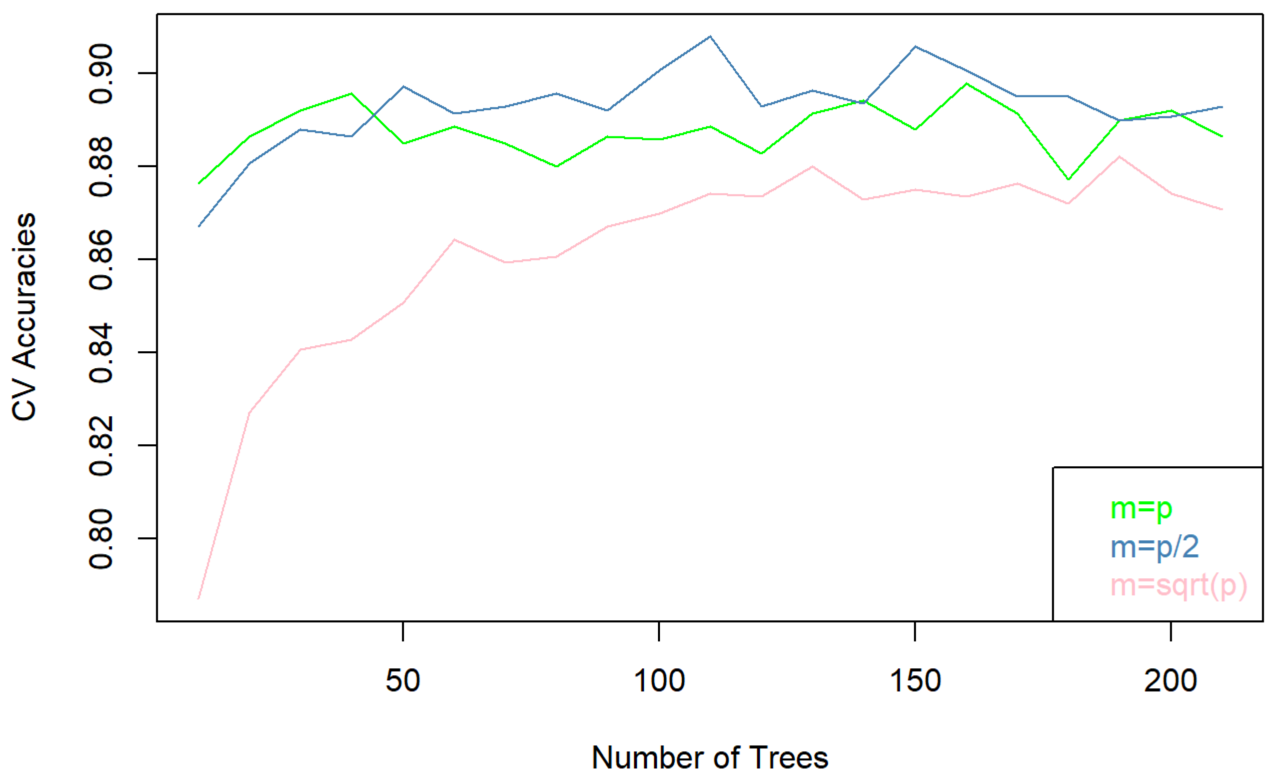
```
mo <- fread('~/R/mobile/train.csv')
mo$price_range <- as.factor(mo$price_range)
mo_obs <- nrow(mo)
mo_idx <- sample(mo_obs, size = trunc(0.70 * mo_obs))
mo_trn <- mo[mo_idx, ]
mo_test <- mo[-mo_idx, ]

Y_test <- mo_test[,price_range]
```

Finding optimal hyperparameter values with K-Fold CV:

```
dd <- data.table() # empty data table to fill with cv accuracies over loop

p <- length(colnames(mo_trn)) -1  # variables for number of predictors hyperparameter
p_ov2 <- p / 2
p_sqrt <- sqrt(length(colnames(mo_trn)) -1)

trees <- seq(from = 10, to = 210, by = 10) # sequence of # of trees hyperparameter

for (num_pred in c(p, p_ov2, p_sqrt)) {
  CV_accuracies = c()
  for (num_trees in trees){

    #Perform K-fold cross validation
    k = 5
    #Randomly shuffle the data
    mo_trn_cross <- mo_trn[sample(nrow(mo_trn)),]

    #Create K equally size folds
    folds <- cut(seq(1,nrow(mo_trn_cross)),breaks=k,labels=FALSE)

    accuracies <- c()

    #Perform K-fold cross validation

    for(i in 1:k){
      #Segement your data by fold using the which() function
      testIndexes <- which(folds==i,arr.ind=TRUE)
      testData <- mo_trn_cross[testIndexes, ]
      trainData <- mo_trn_cross[-testIndexes, ]
      Y_CV <- testData$price_range
      #Use the test and train data partitions however you desire...

      #num_features <- sqrt(length(colnames(mo_trn)) -1)


      rf_classifier <- randomForest(price_range ~ ., data = trainData, ntree = num_trees, mtry =
num_pred, importance = TRUE )
      Y_test_hat <- predict(rf_classifier, newdata = testData, type = "class")

      accuracy <- mean(Y_test_hat == Y_CV)

      accuracies <- c(accuracies, accuracy)
    }
    CV_accuracy <- mean(accuracies)
    CV_accuracies <- c(CV_accuracies, CV_accuracy)
  }
  #print(num_pred)
  dd <- cbind(dd, CV_accuracies)

}
dd[, num_tree := trees]
colnames(dd) <- c("m=p", "m=p/2", "m=sqrt(p)", "num_tree")
dd
```

```
##           m=p     m=p/2 m=sqrt(p) num_tree
##  1: 0.8764286 0.8671429 0.7871429       10
##  2: 0.8864286 0.8807143 0.8271429       20
##  3: 0.8921429 0.8878571 0.8407143       30
##  4: 0.8957143 0.8864286 0.8428571       40
##  5: 0.8850000 0.8971429 0.8507143       50
##  6: 0.8885714 0.8914286 0.8642857       60
##  7: 0.8850000 0.8928571 0.8592857       70
##  8: 0.8800000 0.8957143 0.8607143       80
##  9: 0.8864286 0.8921429 0.8671429       90
## 10: 0.8857143 0.9007143 0.8700000      100
## 11: 0.8885714 0.9078571 0.8742857      110
## 12: 0.8828571 0.8928571 0.8735714      120
## 13: 0.8914286 0.8964286 0.8800000      130
## 14: 0.8942857 0.8935714 0.8728571      140
## 15: 0.8878571 0.9057143 0.8750000      150
## 16: 0.8978571 0.9007143 0.8735714      160
## 17: 0.8914286 0.8950000 0.8764286      170
## 18: 0.8771429 0.8950000 0.8721429      180
## 19: 0.8900000 0.8900000 0.8821429      190
## 20: 0.8921429 0.8907143 0.8742857      200
## 21: 0.8864286 0.8928571 0.8707143      210
##           m=p     m=p/2 m=sqrt(p) num_tree
```

Plotting Accuracy vs Number of Trees for various values of m:



Predictions with Optimal Hyperparameters:

```
rf_classifier <- randomForest(price_range ~ ., data = mo_trn, ntree = 110, mtry = p_ov2, importa
nce = TRUE )

Y_test_hat <- predict(rf_classifier, newdata = mo_test, type = "class")

test_accuracy <- mean(Y_test_hat == Y_test)


cm <- table(observed=Y_test, predicted=Y_test_hat)

diag = diag(cm) # number of correctly classified instances per class
rowsums = apply(cm, 1, sum) # number of instances per class
colsums = apply(cm, 2, sum) # number of predictions per class

precision = diag / colsums
recall = diag / rowsums
f1 = 2 * precision * recall / (precision + recall)

scores <- data.frame(precision, recall, f1)



test_accuracy
```

```
## [1] 0.8933333
```

```
cm
```

```
##         predicted
## observed   0   1   2   3
##        0 139   9   0   0
##        1   9 135   7   0
##        2   0  15 127  16
##        3   0   0   8 135
```

```
scores
```

```
##   precision    recall        f1
## 0 0.9391892 0.9391892 0.9391892
## 1 0.8490566 0.8940397 0.8709677
## 2 0.8943662 0.8037975 0.8466667
## 3 0.8940397 0.9440559 0.9183673
```

Variable Importance:

```
vi <- importance(rf_classifier, type = 2)

vi
```

```
##               MeanDecreaseGini
## battery_power       104.622169
## blue                  1.877017
## clock_speed          12.654598
## dual_sim              2.706942
## fc                   11.300411
## four_g                2.255693
## int_memory           16.358168
## m_dep                12.747777
## mobile_wt            20.929720
## n_cores              10.594363
## pc                   12.307350
## px_height            60.875922
## px_width             58.969501
## ram                 673.496143
## sc_h                 13.673196
## sc_w                 12.554913
## talk_time            14.371397
## three_g               2.221251
## touch_screen          2.182862
## wifi                  2.446852
```