options(tidyverse.quiet=TRUE)  
library(tidyverse)  
library(caret)

## Loading required package: lattice

##   
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':  
##   
## lift

library(ggplot2)  
library(ROCR)  
library(e1071)  
library(mice)

##   
## Attaching package: 'mice'

## The following objects are masked from 'package:base':  
##   
## cbind, rbind

library(VIM)

## Loading required package: colorspace

## Loading required package: grid

## VIM is ready to use.

## Suggestions and bug-reports can be submitted at: https://github.com/statistikat/VIM/issues

##   
## Attaching package: 'VIM'

## The following object is masked from 'package:datasets':  
##   
## sleep

library(ranger)  
library(RColorBrewer)  
library(rattle)

## Loading required package: bitops

## Rattle: A free graphical interface for data science with R.  
## Version 5.4.0 Copyright (c) 2006-2020 Togaware Pty Ltd.  
## Type 'rattle()' to shake, rattle, and roll your data.

##   
## Attaching package: 'rattle'

## The following object is masked from 'package:ranger':  
##   
## importance

## The following object is masked from 'package:VIM':  
##   
## wine

library(rpart)  
library(leaps)  
library(GGally)

## Registered S3 method overwritten by 'GGally':  
## method from   
## +.gg ggplot2

##   
## Attaching package: 'GGally'

## The following object is masked from 'package:dplyr':  
##   
## nasa

library(gridExtra)

##   
## Attaching package: 'gridExtra'

## The following object is masked from 'package:dplyr':  
##   
## combine

library(car)

## Loading required package: carData

##   
## Attaching package: 'car'

## The following object is masked from 'package:dplyr':  
##   
## recode

## The following object is masked from 'package:purrr':  
##   
## some

library(lmtest)

## Loading required package: zoo

##   
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':  
##   
## as.Date, as.Date.numeric

library(ggcorrplot)  
library(lubridate)

##   
## Attaching package: 'lubridate'

## The following objects are masked from 'package:dplyr':  
##   
## intersect, setdiff, union

## The following objects are masked from 'package:base':  
##   
## date, intersect, setdiff, union

library(cluster)  
library(factoextra)

## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa

library(dendextend)

##   
## ---------------------  
## Welcome to dendextend version 1.13.4  
## Type citation('dendextend') for how to cite the package.  
##   
## Type browseVignettes(package = 'dendextend') for the package vignette.  
## The github page is: https://github.com/talgalili/dendextend/  
##   
## Suggestions and bug-reports can be submitted at: https://github.com/talgalili/dendextend/issues  
## Or contact: <tal.galili@gmail.com>  
##   
## To suppress this message use: suppressPackageStartupMessages(library(dendextend))  
## ---------------------

##   
## Attaching package: 'dendextend'

## The following object is masked from 'package:rpart':  
##   
## prune

## The following object is masked from 'package:stats':  
##   
## cutree

library(readr)  
chicago <- read\_csv("chicago.csv")

## Warning: Missing column names filled in: 'X1' [1]

## Parsed with column specification:  
## cols(  
## .default = col\_character(),  
## X1 = col\_double(),  
## ID = col\_double(),  
## Arrest = col\_logical(),  
## Domestic = col\_logical(),  
## Ward = col\_double(),  
## `Community Area` = col\_double(),  
## `X Coordinate` = col\_double(),  
## `Y Coordinate` = col\_double(),  
## Year = col\_double(),  
## Latitude = col\_double(),  
## Longitude = col\_double()  
## )

## See spec(...) for full column specifications.

chicago = chicago %>% drop\_na()

chicago = chicago %>% select(-X1, -ID, -Longitude, -Latitude, -`Case Number`, -Location, -`Updated On`, -`X Coordinate`, -`Y Coordinate`) %>%  
 mutate(Block = as.factor(Block)) %>%  
 mutate(IUCR = as.factor(IUCR)) %>%  
 mutate(`Primary Type` = as.factor(`Primary Type`)) %>%  
 mutate(Description = as.factor(Description)) %>%  
 mutate(`Location Description` = as.factor(`Location Description`)) %>%  
 mutate(Arrest = as.factor(Arrest)) %>%  
 mutate(Arrest = fct\_recode(Arrest, "False" = "0", "True" = "1")) %>%  
 mutate(Domestic = as.factor(Domestic)) %>%  
 mutate(Domestic = fct\_recode(Domestic, "False" = "0", "True" = "1")) %>%  
 mutate(Beat = as.factor(Beat)) %>%  
 mutate(District = as.factor(District)) %>%  
 mutate(`FBI Code` = as.factor(`FBI Code`)) %>%  
 mutate(Date = as.character.Date(Date))

## Warning: Unknown levels in `f`: 0, 1  
  
## Warning: Unknown levels in `f`: 0, 1

Removed unnecessary variables from data frame “chicago” and converted some remaining variables into factors.

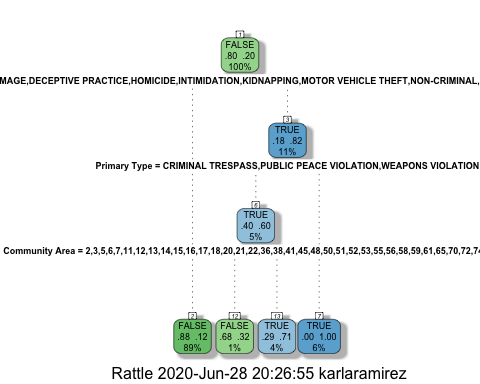
chicago = chicago %>% mutate (Ward = as.factor(Ward)) %>% mutate(`Community Area` = as.factor(`Community Area`))

chicago\_small = chicago %>% select(`Primary Type`, Arrest, Domestic, District, Ward, `Community Area`, `FBI Code`)

Logistic Regression Models

set.seed(1234)  
train.rows = createDataPartition(y = chicago\_small$Arrest, p=0.7, list = FALSE)  
train = slice(chicago\_small, train.rows)  
test = slice(chicago\_small, -train.rows)

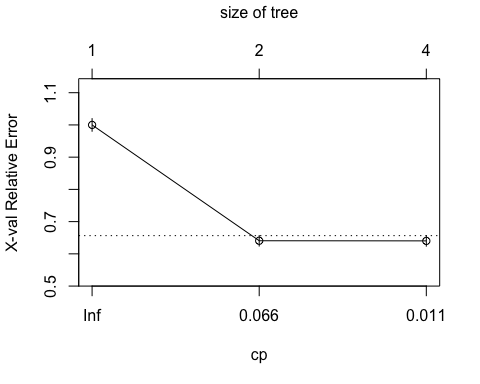
smalltree1 = rpart(Arrest~., method="class", train)  
fancyRpartPlot(smalltree1)



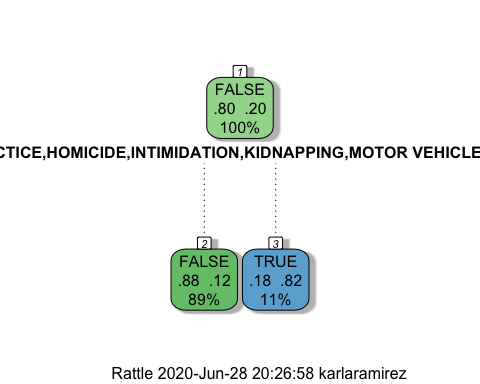
printcp(smalltree1)

##   
## Classification tree:  
## rpart(formula = Arrest ~ ., data = train, method = "class")  
##   
## Variables actually used in tree construction:  
## [1] Community Area Primary Type   
##   
## Root node error: 2087/10300 = 0.20262  
##   
## n= 10300   
##   
## CP nsplit rel error xerror xstd  
## 1 0.359847 0 1.00000 1.00000 0.019547  
## 2 0.012218 1 0.64015 0.64063 0.016344  
## 3 0.010000 3 0.61572 0.64015 0.016339

plotcp(smalltree1)



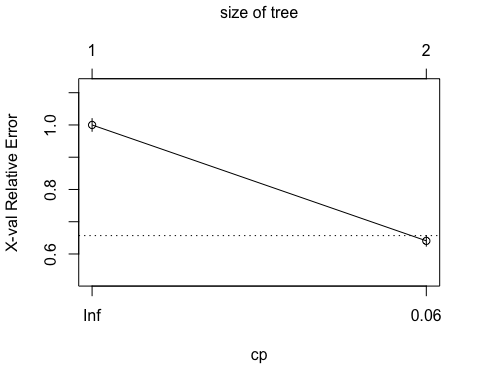
smalltree2 = rpart(Arrest ~ `Primary Type` + District, method="class", train)  
fancyRpartPlot(smalltree2)



printcp(smalltree2)

##   
## Classification tree:  
## rpart(formula = Arrest ~ `Primary Type` + District, data = train,   
## method = "class")  
##   
## Variables actually used in tree construction:  
## [1] Primary Type  
##   
## Root node error: 2087/10300 = 0.20262  
##   
## n= 10300   
##   
## CP nsplit rel error xerror xstd  
## 1 0.35985 0 1.00000 1.00000 0.019547  
## 2 0.01000 1 0.64015 0.64063 0.016344

plotcp(smalltree2)



treepred1 = predict(smalltree1, train, type = "class")  
head(treepred1)

## 1 2 3 4 5 6   
## FALSE FALSE FALSE FALSE FALSE FALSE   
## Levels: FALSE TRUE

confusionMatrix(treepred1,train$Arrest,positive="TRUE")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction FALSE TRUE  
## FALSE 8101 1173  
## TRUE 112 914  
##   
## Accuracy : 0.8752   
## 95% CI : (0.8687, 0.8816)  
## No Information Rate : 0.7974   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.5236   
##   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Sensitivity : 0.43795   
## Specificity : 0.98636   
## Pos Pred Value : 0.89084   
## Neg Pred Value : 0.87352   
## Prevalence : 0.20262   
## Detection Rate : 0.08874   
## Detection Prevalence : 0.09961   
## Balanced Accuracy : 0.71216   
##   
## 'Positive' Class : TRUE   
##

treepred\_test1 = predict(smalltree1, test, type = "class")  
head(treepred\_test1)

## 1 2 3 4 5 6   
## FALSE FALSE FALSE FALSE FALSE FALSE   
## Levels: FALSE TRUE

confusionMatrix(treepred\_test1,test$Arrest,positive="TRUE")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction FALSE TRUE  
## FALSE 3471 501  
## TRUE 48 393  
##   
## Accuracy : 0.8756   
## 95% CI : (0.8655, 0.8852)  
## No Information Rate : 0.7974   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.5252   
##   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Sensitivity : 0.43960   
## Specificity : 0.98636   
## Pos Pred Value : 0.89116   
## Neg Pred Value : 0.87387   
## Prevalence : 0.20258   
## Detection Rate : 0.08906   
## Detection Prevalence : 0.09993   
## Balanced Accuracy : 0.71298   
##   
## 'Positive' Class : TRUE   
##

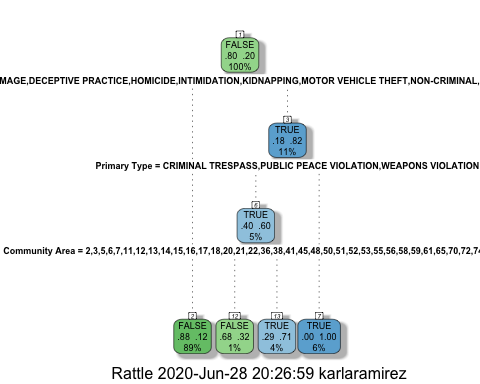
treepred2 = predict(smalltree2, train, type = "class")  
confusionMatrix(treepred2,train$Arrest,positive="TRUE")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction FALSE TRUE  
## FALSE 8003 1126  
## TRUE 210 961  
##   
## Accuracy : 0.8703   
## 95% CI : (0.8636, 0.8767)  
## No Information Rate : 0.7974   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.52   
##   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Sensitivity : 0.4605   
## Specificity : 0.9744   
## Pos Pred Value : 0.8207   
## Neg Pred Value : 0.8767   
## Prevalence : 0.2026   
## Detection Rate : 0.0933   
## Detection Prevalence : 0.1137   
## Balanced Accuracy : 0.7175   
##   
## 'Positive' Class : TRUE   
##

treepred\_test2 = predict(smalltree2, test, type = "class")  
confusionMatrix(treepred\_test2,test$Arrest,positive="TRUE")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction FALSE TRUE  
## FALSE 3447 472  
## TRUE 72 422  
##   
## Accuracy : 0.8767   
## 95% CI : (0.8667, 0.8863)  
## No Information Rate : 0.7974   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.542   
##   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Sensitivity : 0.47204   
## Specificity : 0.97954   
## Pos Pred Value : 0.85425   
## Neg Pred Value : 0.87956   
## Prevalence : 0.20258   
## Detection Rate : 0.09563   
## Detection Prevalence : 0.11194   
## Balanced Accuracy : 0.72579   
##   
## 'Positive' Class : TRUE   
##

smalltree1 = prune(smalltree1,cp= smalltree1$cptable[which.min(smalltree1$cptable[,"xerror"]),"CP"])  
fancyRpartPlot(smalltree1)



Random Forests

treepred1 = predict(smalltree1, train, type = "class")  
confusionMatrix(treepred1,train$Arrest,positive="TRUE")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction FALSE TRUE  
## FALSE 8101 1173  
## TRUE 112 914  
##   
## Accuracy : 0.8752   
## 95% CI : (0.8687, 0.8816)  
## No Information Rate : 0.7974   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.5236   
##   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Sensitivity : 0.43795   
## Specificity : 0.98636   
## Pos Pred Value : 0.89084   
## Neg Pred Value : 0.87352   
## Prevalence : 0.20262   
## Detection Rate : 0.08874   
## Detection Prevalence : 0.09961   
## Balanced Accuracy : 0.71216   
##   
## 'Positive' Class : TRUE   
##

treepred\_test1 = predict(smalltree1, test, type = "class")  
confusionMatrix(treepred\_test1,test$Arrest,positive="TRUE")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction FALSE TRUE  
## FALSE 3471 501  
## TRUE 48 393  
##   
## Accuracy : 0.8756   
## 95% CI : (0.8655, 0.8852)  
## No Information Rate : 0.7974   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.5252   
##   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Sensitivity : 0.43960   
## Specificity : 0.98636   
## Pos Pred Value : 0.89116   
## Neg Pred Value : 0.87387   
## Prevalence : 0.20258   
## Detection Rate : 0.08906   
## Detection Prevalence : 0.09993   
## Balanced Accuracy : 0.71298   
##   
## 'Positive' Class : TRUE   
##

train\_under50 = train %>% select(-`Community Area`)

test\_under50 = test %>% select(-`Community Area`)

I realized I was getting errors for my Random Forests because the maximum levels it would handle were 53 and Community Area had 77, so I removed that variable and created a new train/test set labled “under 50”.

fit\_control = trainControl(method = "cv", number = 10)  
  
set.seed(1234)  
rf\_fit = train(x=as.matrix(train\_under50[,-2]), y=as.matrix(train\_under50$Arrest),   
 method = "ranger",   
 importance = "permutation",  
 trControl = fit\_control)

#saveRDS(rf\_fit, "rf\_fit.rds")

#rf\_fit = readRDS("rf\_fit.rds")

varImp(rf\_fit)

## ranger variable importance  
##   
## Overall  
## FBI Code 100.000  
## Primary Type 39.929  
## District 4.831  
## Ward 4.662  
## Domestic 0.000

rf\_fit

## Random Forest   
##   
## 10300 samples  
## 5 predictor  
## 2 classes: 'FALSE', 'TRUE'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 9270, 9270, 9269, 9269, 9269, 9270, ...   
## Resampling results across tuning parameters:  
##   
## mtry splitrule Accuracy Kappa   
## 2 gini 0.8205869 0.3987246  
## 2 extratrees 0.8147515 0.1710134  
## 3 gini 0.8153439 0.3917617  
## 3 extratrees 0.8137798 0.2067673  
## 5 gini 0.8119431 0.3780088  
## 5 extratrees 0.8050422 0.2273430  
##   
## Tuning parameter 'min.node.size' was held constant at a value of 1  
## Accuracy was used to select the optimal model using the largest value.  
## The final values used for the model were mtry = 2, splitrule = gini  
## and min.node.size = 1.

predRF = predict(rf\_fit)  
head(predRF)

## [1] FALSE FALSE FALSE FALSE FALSE FALSE  
## Levels: FALSE TRUE

confusionMatrix(predRF, train\_under50$Arrest, positive = "TRUE")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction FALSE TRUE  
## FALSE 8085 1096  
## TRUE 128 991  
##   
## Accuracy : 0.8812   
## 95% CI : (0.8748, 0.8874)  
## No Information Rate : 0.7974   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.5553   
##   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Sensitivity : 0.47484   
## Specificity : 0.98441   
## Pos Pred Value : 0.88561   
## Neg Pred Value : 0.88062   
## Prevalence : 0.20262   
## Detection Rate : 0.09621   
## Detection Prevalence : 0.10864   
## Balanced Accuracy : 0.72963   
##   
## 'Positive' Class : TRUE   
##

predRF\_test = predict(rf\_fit, newdata = test\_under50)

confusionMatrix(predRF\_test, test\_under50$Arrest, positive = "TRUE")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction FALSE TRUE  
## FALSE 3458 490  
## TRUE 61 404  
##   
## Accuracy : 0.8751   
## 95% CI : (0.865, 0.8848)  
## No Information Rate : 0.7974   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.5293   
##   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Sensitivity : 0.45190   
## Specificity : 0.98267   
## Pos Pred Value : 0.86882   
## Neg Pred Value : 0.87589   
## Prevalence : 0.20258   
## Detection Rate : 0.09155   
## Detection Prevalence : 0.10537   
## Balanced Accuracy : 0.71728   
##   
## 'Positive' Class : TRUE   
##

The training set in a random forest was slightly better than the testing set by 1%.

Parameter Tuning

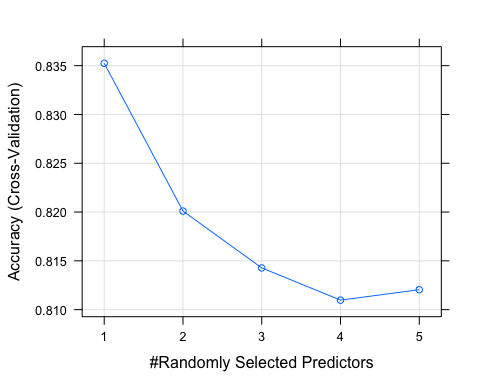
tunegrid = expand.grid(mtry = 1:5, splitrule = c("hellinger"), min.node.size=1)

set.seed(1234)   
rf\_fit2 = train(x = as.matrix(train\_under50[,-2]),y = as.matrix(train\_under50$Arrest),  
 method = "ranger",   
 tuneGrid = tunegrid,  
 importance = "permutation",   
 trControl = fit\_control)

print(rf\_fit2)

## Random Forest   
##   
## 10300 samples  
## 5 predictor  
## 2 classes: 'FALSE', 'TRUE'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 9270, 9270, 9269, 9269, 9269, 9270, ...   
## Resampling results across tuning parameters:  
##   
## mtry Accuracy Kappa   
## 1 0.8352462 0.4143868  
## 2 0.8201009 0.3977611  
## 3 0.8142758 0.3897378  
## 4 0.8109748 0.3841099  
## 5 0.8120407 0.3778430  
##   
## Tuning parameter 'splitrule' was held constant at a value of hellinger  
##   
## Tuning parameter 'min.node.size' was held constant at a value of 1  
## Accuracy was used to select the optimal model using the largest value.  
## The final values used for the model were mtry = 1, splitrule = hellinger  
## and min.node.size = 1.

plot(rf\_fit2)



varImp(rf\_fit2)

## ranger variable importance  
##   
## Overall  
## FBI Code 100.000  
## Primary Type 40.494  
## Domestic 3.346  
## District 1.268  
## Ward 0.000

predRF2 = predict(rf\_fit2, train\_under50, type = "raw")  
confusionMatrix(predRF2, train\_under50$Arrest, positive = "TRUE")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction FALSE TRUE  
## FALSE 8107 1228  
## TRUE 106 859  
##   
## Accuracy : 0.8705   
## 95% CI : (0.8638, 0.8769)  
## No Information Rate : 0.7974   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.4987   
##   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Sensitivity : 0.41160   
## Specificity : 0.98709   
## Pos Pred Value : 0.89016   
## Neg Pred Value : 0.86845   
## Prevalence : 0.20262   
## Detection Rate : 0.08340   
## Detection Prevalence : 0.09369   
## Balanced Accuracy : 0.69934   
##   
## 'Positive' Class : TRUE   
##

predRFtest2 = predict(rf\_fit2, newdata = test\_under50, type = "raw")  
confusionMatrix(predRFtest2, test\_under50$Arrest, positive = "TRUE")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction FALSE TRUE  
## FALSE 3489 518  
## TRUE 30 376  
##   
## Accuracy : 0.8758   
## 95% CI : (0.8657, 0.8854)  
## No Information Rate : 0.7974   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.5174   
##   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Sensitivity : 0.4206   
## Specificity : 0.9915   
## Pos Pred Value : 0.9261   
## Neg Pred Value : 0.8707   
## Prevalence : 0.2026   
## Detection Rate : 0.0852   
## Detection Prevalence : 0.0920   
## Balanced Accuracy : 0.7060   
##   
## 'Positive' Class : TRUE   
##

The training set in rf\_fit2 did a little better than the training set in rf\_fit when using parameter tuning. I set the tunegrid as hellinger because it is a split rule designed to fix imbalanced data. While the observation amounts are the same, I was concerned about the different levels between each variable. Not sure if this helped much but I’ll use the gini split rule for my next predictions along with the optimal mtry amount in the tunegrid since both rules had higher accuracy scores in my previous predictions.

tunegrid2 = expand.grid(mtry = 1, splitrule = c("gini"), min.node.size=1)

set.seed(1234)  
rf\_fit3 = train(x = as.matrix(train\_under50[,-2]),y = as.matrix(train\_under50$Arrest),  
 method = "ranger",  
 tuneGrid = tunegrid2,  
 importance = "permutation",  
 max.depth = 5,  
 trControl = fit\_control)

print(rf\_fit3)

## Random Forest   
##   
## 10300 samples  
## 5 predictor  
## 2 classes: 'FALSE', 'TRUE'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 9270, 9270, 9269, 9269, 9269, 9270, ...   
## Resampling results:  
##   
## Accuracy Kappa   
## 0.8336946 0.3830779  
##   
## Tuning parameter 'mtry' was held constant at a value of 1  
## Tuning  
## parameter 'splitrule' was held constant at a value of gini  
## Tuning  
## parameter 'min.node.size' was held constant at a value of 1

#plot(rf\_fit3)  
varImp(rf\_fit3)

## ranger variable importance  
##   
## Overall  
## FBI Code 100.0000  
## Primary Type 29.7654  
## Domestic 3.7166  
## Ward 0.7748  
## District 0.0000

predRF3 = predict(rf\_fit3, train\_under50, type = "raw")  
confusionMatrix(predRF3, train\_under50$Arrest, positive = "TRUE")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction FALSE TRUE  
## FALSE 8129 1386  
## TRUE 84 701  
##   
## Accuracy : 0.8573   
## 95% CI : (0.8504, 0.864)  
## No Information Rate : 0.7974   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.4244   
##   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Sensitivity : 0.33589   
## Specificity : 0.98977   
## Pos Pred Value : 0.89299   
## Neg Pred Value : 0.85434   
## Prevalence : 0.20262   
## Detection Rate : 0.06806   
## Detection Prevalence : 0.07621   
## Balanced Accuracy : 0.66283   
##   
## 'Positive' Class : TRUE   
##

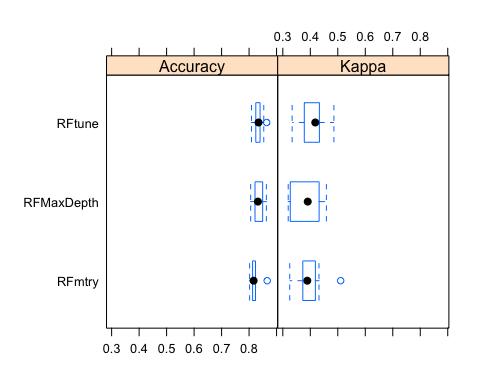
predRFtest3 = predict(rf\_fit3, newdata = test\_under50, type = "raw")  
confusionMatrix(predRFtest3, test\_under50$Arrest, positive = "TRUE")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction FALSE TRUE  
## FALSE 3495 563  
## TRUE 24 331  
##   
## Accuracy : 0.867   
## 95% CI : (0.8566, 0.8769)  
## No Information Rate : 0.7974   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.4689   
##   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Sensitivity : 0.37025   
## Specificity : 0.99318   
## Pos Pred Value : 0.93239   
## Neg Pred Value : 0.86126   
## Prevalence : 0.20258   
## Detection Rate : 0.07501   
## Detection Prevalence : 0.08044   
## Balanced Accuracy : 0.68171   
##   
## 'Positive' Class : TRUE   
##

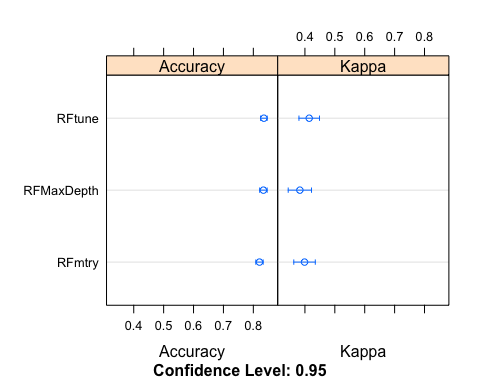
This prediction was surprising… looks like updating the tunegrid for rf\_fit3 made the testing set more accurate than the training set.

Comparing all 3:

results = resamples(list(RFmtry=rf\_fit, RFtune=rf\_fit2, RFMaxDepth=rf\_fit3))  
# boxplots of results  
bwplot(results)



# dotplots of results  
dotplot(results)

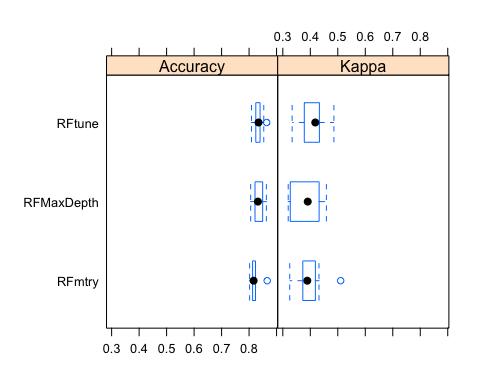


#saveRDS(rf\_fit2, "rf\_fit2.rds")  
#rm(rf\_fit2)  
#saveRDS(rf\_fit3, "rf\_fit3.rds")  
#rm(rf\_fit3)

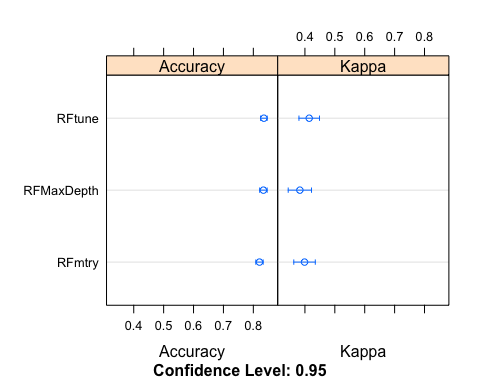
rf\_fit2 = readRDS("rf\_fit2.rds")  
rf\_fit3 = readRDS("rf\_fit3.rds")

Comparing another set of 3

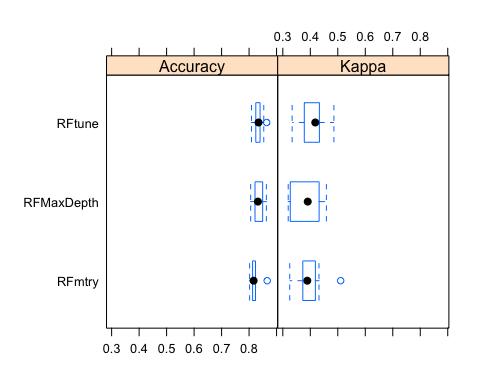
results2 = resamples(list(RFmtry=rf\_fit2, RFtune=rf\_fit3, RFMaxDepth=rf\_fit))  
# boxplots of results  
bwplot(results)



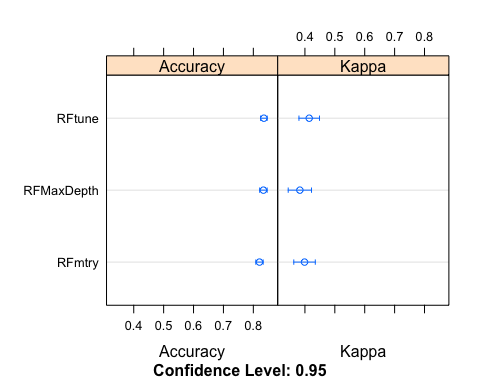
# dotplots of results  
dotplot(results)



results3 = resamples(list(RFmtry=rf\_fit3, RFtune=rf\_fit3, RFMaxDepth=rf\_fit3))  
# boxplots of results  
bwplot(results)



# dotplots of results  
dotplot(results)



fit\_control2 = trainControl(method = "cv", number = 5)

set.seed(1234)  
rf\_fit4 = train(x = as.matrix(train\_under50[,-2]),y = as.matrix(train\_under50$Arrest),  
 method = "ranger",  
 tuneGrid = tunegrid2,  
 importance = "permutation",  
 max.depth = 5,  
 trControl = fit\_control2)

print(rf\_fit3)

## Random Forest   
##   
## 10300 samples  
## 5 predictor  
## 2 classes: 'FALSE', 'TRUE'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 9270, 9270, 9269, 9269, 9269, 9270, ...   
## Resampling results:  
##   
## Accuracy Kappa   
## 0.8336946 0.3830779  
##   
## Tuning parameter 'mtry' was held constant at a value of 1  
## Tuning  
## parameter 'splitrule' was held constant at a value of gini  
## Tuning  
## parameter 'min.node.size' was held constant at a value of 1

varImp(rf\_fit3)

## ranger variable importance  
##   
## Overall  
## FBI Code 100.0000  
## Primary Type 29.7654  
## Domestic 3.7166  
## Ward 0.7748  
## District 0.0000

predRF4 = predict(rf\_fit4, train\_under50, type = "raw")  
confusionMatrix(predRF4, train\_under50$Arrest, positive = "TRUE")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction FALSE TRUE  
## FALSE 8151 1410  
## TRUE 62 677  
##   
## Accuracy : 0.8571   
## 95% CI : (0.8502, 0.8638)  
## No Information Rate : 0.7974   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.4174   
##   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Sensitivity : 0.32439   
## Specificity : 0.99245   
## Pos Pred Value : 0.91610   
## Neg Pred Value : 0.85253   
## Prevalence : 0.20262   
## Detection Rate : 0.06573   
## Detection Prevalence : 0.07175   
## Balanced Accuracy : 0.65842   
##   
## 'Positive' Class : TRUE   
##

I created rf\_fit4 because I was trying get better sensitivity results. I tried adjusting the trControl function to have a cross-validation of 5 vs 10 but the sensitivity results came out worse. So far, out of all my predictions, the original rf\_fit has come out with the best results overall in terms of accuracy, sensitivity, and specificity.

#saveRDS(rf\_fit4, "rf\_fit4.rds")  
#rm(rf\_fit4)

rf\_fit4 = readRDS("rf\_fit4.rds")