# BAN502 Final Project

## Kyle Capponcelli

### Project Phase 2

Libraries

library("tidyverse", quietly = TRUE, warn.conflicts = FALSE)

## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --

## v ggplot2 3.2.1 v purrr 0.3.3  
## v tibble 2.1.3 v dplyr 0.8.3  
## v tidyr 1.0.0 v stringr 1.4.0  
## v readr 1.3.1 v forcats 0.4.0

## -- Conflicts ------------------------------------------ tidyverse\_conflicts() --  
## x dplyr::filter() masks stats::filter()  
## x dplyr::lag() masks stats::lag()

library("corrplot")

## corrplot 0.84 loaded

library("lubridate")

##   
## Attaching package: 'lubridate'

## The following object is masked from 'package:base':  
##   
## date

library("lmtest")

## Loading required package: zoo

##   
## Attaching package: 'zoo'

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

library("leaps")  
library("lme4")

## Loading required package: Matrix

##   
## Attaching package: 'Matrix'

## The following objects are masked from 'package:tidyr':  
##   
## expand, pack, unpack

library("e1071")  
library("caret")

## Loading required package: lattice

##   
## Attaching package: 'caret'

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

library("ROCR")

## Loading required package: gplots

##   
## Attaching package: 'gplots'

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

library("ggcorrplot")  
library("GGally")

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

##   
## Attaching package: 'GGally'

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

Data Load

chicago <- read\_csv("chicago.csv", col\_types = cols(Date = col\_datetime(format = "%m/%d/%Y %H:%M")))  
View(chicago)

Changed the variable of Date in excel file during read-in to fix issues with later date conversion as seen in Phase 1.

Column-wise Deletion

chicago = chicago %>% select(-"ID", -"Case Number", -"Updated On", -"X Coordinate", - "Y Coordinate", -"Location", -"Latitude", -"Longitude")

Drop NAs

chicago = drop\_na(chicago)  
#summary(chicago)

Time Conversion

chicago = chicago %>% mutate(Hour = hour(Date))  
  
chicago = chicago %>% mutate(Month = month(Date))  
  
chicago = chicago %>% mutate(Minute = minute(Date))  
  
chicago = chicago %>% mutate(Day = day(Date))

Removed the code to convert entire date; chicago = chicago %>% mutate(Date = mdy\_hms(Date)), in order to conduct more detailed analysis.

Factor Conversion

chicago = chicago %>% mutate(Arrest = as\_factor(as.character(Arrest)))  
   
chicago = chicago %>% mutate(IUCR = as\_factor(IUCR))  
  
chicago = chicago %>% mutate(Beat = as\_factor(Beat))  
  
chicago = chicago %>% mutate(Ward = as\_factor(Ward))  
  
chicago = chicago %>% mutate(District = as\_factor(District))  
  
chicago = chicago %>% mutate(`Primary Type` = as\_factor(`Primary Type`))  
  
chicago = chicago %>% mutate(Description = as\_factor(Description))  
  
chicago = chicago %>% mutate(`Location Description` = as\_factor(`Location Description`))  
  
chicago = chicago %>% mutate(`FBI Code` = as\_factor(as.character(`FBI Code`))) %>%  
 mutate(`FBI Code` = fct\_recode(`FBI Code`, "Criminal Sexual Assault" = "2", "Fraud" = "11", "Criminal Sexual Abuse" = "17", "Misc Non-Index Offense" = "26", "Simple Battery" = "08B", "Vandalism"= "14", "Motor Vehicle Theft" = "7", "Larceny" = "6", "Offenses Against Family" = "20", "Simple Assault" = "08A", "Aggravated Assault" = "04A", "Burglary" = "5", "Aggravated Assault" = "04A", "Weapons Violation" = "15", "Drug Abuse" = "18", "Robbery" = "3", "Liquor License" = "22", "Homicide" = "01A", "Forgery and Counterfeiting" = "10", "Disorderly Conduct" = "24", "Embezzlement" = "12", "Arson" = "9", "Gambling" = "19", "Prostitution" = "16", "Involuntary Manslaughter" = "01B", "Stolen Property" = "13"))  
  
chicago = chicago %>% mutate(Hour = as\_factor(Hour))  
  
chicago = chicago %>% mutate(Month = as\_factor(as.character(Month)))%>%  
 mutate(Month = fct\_recode(Month, "January" = "1",  
 "February" = "2","March" = "3","April" = "4","May" = "5","June" = "6","July" = "7","August" = "8","September" = "9","October" = "10","November" = "11","December" = "12"))  
  
chicago = chicago %>% mutate(Minute = as\_factor(Minute))  
  
chicago = chicago %>% mutate(Day = as\_factor(Day))

Added Beat, Ward and District to list of conversion to factors.

Due to corrections within the date/time functions, time may now be significant and may be used as a predictor of Arrests, issue with dates has been fixed since Phase\_1 of the project.

Grouping

chicago %>% group\_by(`Primary Type`,IUCR, District)

## # A tibble: 266,236 x 18  
## # Groups: Primary Type, IUCR, District [4,541]  
## Date Block IUCR `Primary Type` Description `Location Descr~  
## <dttm> <chr> <fct> <fct> <fct> <fct>   
## 1 2018-01-01 00:00:00 069X~ 1753 OFFENSE INVOL~ SEX ASSLT ~ RESIDENCE-GARAGE  
## 2 2018-01-01 00:00:00 070X~ 1130 DECEPTIVE PRA~ FRAUD OR C~ APARTMENT   
## 3 2018-01-01 00:00:00 072X~ 1153 DECEPTIVE PRA~ FINANCIAL ~ RESIDENCE   
## 4 2018-01-01 00:00:00 047X~ 1752 OFFENSE INVOL~ AGG CRIM S~ RESIDENCE   
## 5 2018-01-01 00:00:00 051X~ 1752 OFFENSE INVOL~ AGG CRIM S~ RESIDENCE   
## 6 2018-01-01 00:00:00 013X~ 265 CRIM SEXUAL A~ AGGRAVATED~ OTHER   
## 7 2018-01-01 00:00:00 017X~ 281 CRIM SEXUAL A~ NON-AGGRAV~ OTHER   
## 8 2018-01-01 00:00:00 056X~ 1752 OFFENSE INVOL~ AGG CRIM S~ RESIDENCE   
## 9 2018-01-01 00:00:00 009X~ 1752 OFFENSE INVOL~ AGG CRIM S~ APARTMENT   
## 10 2018-01-01 00:00:00 052X~ 2826 OTHER OFFENSE HARASSMENT~ RESIDENCE   
## # ... with 266,226 more rows, and 12 more variables: Arrest <fct>,  
## # Domestic <lgl>, Beat <fct>, District <fct>, Ward <fct>, `Community  
## # Area` <dbl>, `FBI Code` <fct>, Year <dbl>, Hour <fct>, Month <fct>,  
## # Minute <fct>, Day <fct>

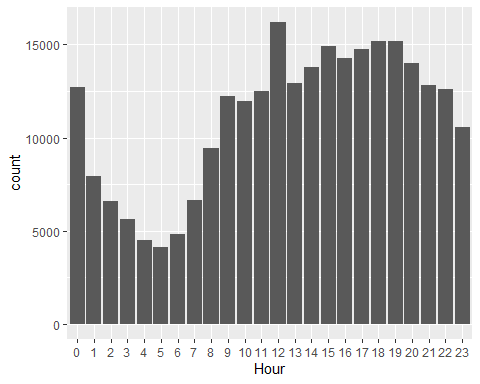
#### Time Histogram

Below is the Hour variable that I fixed from Phase 1.

Hour

ggplot(chicago, aes(x=Hour)) + geom\_histogram(stat = "count")

## Warning: Ignoring unknown parameters: binwidth, bins, pad

 Hour appears as though it may be a predictor of Arrest given the variability in the histogram. I will use this as a predictor in my model.

Variables to use in Model:

Description and IUCR have been dropped to large amount of variables in each item, many with little or no data points. Hour has been added.

Arrest ~ Primary Type District FBI Code Hour

chicago1 = chicago %>% select("Arrest", "Primary Type", "District", "FBI Code", "Hour")  
# summary(chicago1)

Data Split

set.seed(1234)  
train.rows = createDataPartition(y = chicago1$Arrest, p=0.7, list = FALSE)  
train = chicago1[train.rows,]  
test = chicago1[-train.rows,]

#### Model Notes

All models will use a positive class of TRUE, meaning that the [TRUE, TRUE] category (sensitivity) is that an Arrest was predicted and an Arrest was made.

#### Random Forests

Random Forests

# fit\_control = trainControl(method = "cv",  
# number = 10)   
#   
# set.seed(1234)  
# rf\_fit = train(x=as.matrix(train[,-1]), y=as.matrix(train$Arrest),  
# method = "ranger",  
# importance = "permutation",  
# trControl = fit\_control)

Save Model

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

Load Model

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

Random Forest Details

varImp(rf\_fit)

## ranger variable importance  
##   
## Overall  
## FBI Code 100.0000  
## Primary Type 76.5464  
## District 0.6079  
## Hour 0.0000

rf\_fit

## Random Forest   
##   
## 186366 samples  
## 4 predictor  
## 2 classes: 'FALSE', 'TRUE'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 167729, 167730, 167729, 167729, 167729, 167730, ...   
## Resampling results across tuning parameters:  
##   
## mtry splitrule Accuracy Kappa   
## 2 gini 0.8391391 0.4068853  
## 2 extratrees 0.8376688 0.3486095  
## 3 gini 0.8483198 0.4477196  
## 3 extratrees 0.8358498 0.3471952  
## 4 gini 0.8433563 0.4350022  
## 4 extratrees 0.8302854 0.3451261  
##   
## 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 = 3, splitrule = gini  
## and min.node.size = 1.

Predictions Train

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

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

Confusion matrix Train

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

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction FALSE TRUE  
## FALSE 146882 20399  
## TRUE 2093 16992  
##   
## Accuracy : 0.8793   
## 95% CI : (0.8778, 0.8808)  
## No Information Rate : 0.7994   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.5393   
##   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Sensitivity : 0.45444   
## Specificity : 0.98595   
## Pos Pred Value : 0.89033   
## Neg Pred Value : 0.87806   
## Prevalence : 0.20063   
## Detection Rate : 0.09118   
## Detection Prevalence : 0.10241   
## Balanced Accuracy : 0.72020   
##   
## 'Positive' Class : TRUE   
##

Accuracy on this model is shown to be significantly better than the null accuracy. Sensitivity is not particularly good but this model shows that it is still more accurate than null.

Predictions Test

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

Confusion Matrix Test

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

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction FALSE TRUE  
## FALSE 35091 8664  
## TRUE 28755 7360  
##   
## Accuracy : 0.5315   
## 95% CI : (0.528, 0.535)  
## No Information Rate : 0.7994   
## P-Value [Acc > NIR] : 1   
##   
## Kappa : 0.0061   
##   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 0.45931   
## Specificity : 0.54962   
## Pos Pred Value : 0.20379   
## Neg Pred Value : 0.80199   
## Prevalence : 0.20063   
## Detection Rate : 0.09215   
## Detection Prevalence : 0.45217   
## Balanced Accuracy : 0.50447   
##   
## 'Positive' Class : TRUE   
##

The drastic drop in accuracy for this model is concerning. I will most likely not recommend this model for future use.

#### K-Fold Cross Validation

Model Creation

# kctrl = trainControl(method = "cv",number = 10)  
#   
# set.seed(1234)  
# modkFold = train(Arrest ~., chicago1, method = "glm", trControl = kctrl)  
# summary(modkFold)

Save Model

# saveRDS(modkFold, "modkFold.rds")

Load Model

modkFold = readRDS("modkFold.rds")

Predictions

predictions = predict(modkFold, type="prob")[,2]

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :  
## prediction from a rank-deficient fit may be misleading

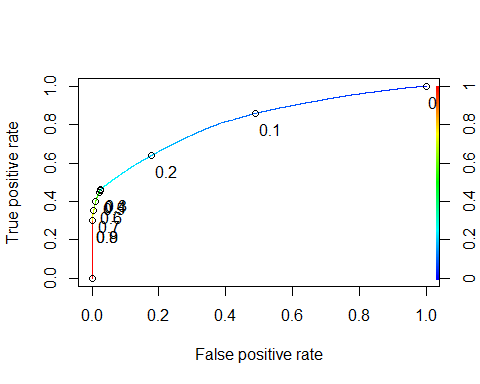
head(predictions)

## [1] 0.13061142 0.03637839 0.03214607 0.04900596 0.05500105 0.08128112

Given the misleading fits I will most likely not use this model.

Threshold Selection

ROCRpred = prediction(predictions, chicago1$Arrest)  
  
ROCRperf = performance(ROCRpred, "tpr", "fpr")  
plot(ROCRperf, colorize=TRUE, print.cutoffs.at=seq(0,1,by=0.1), text.adj=c(-0.2,1.7))

 Given the low AUC and the left hand side of the curve being far below 1 I would say that this model is not functioning well. This makes sense since negative impact warnings were shown throughout its creation.

as.numeric(performance(ROCRpred, "auc")@y.values)

## [1] 0.8136976

Optimal Cut-off

opt.cut = function(perf, pred){  
 cut.ind = mapply(FUN=function(x, y, p){  
 d = (x - 0)^2 + (y-1)^2  
 ind = which(d == min(d))  
 c(sensitivity = y[[ind]], specificity = 1-x[[ind]],   
 cutoff = p[[ind]])  
 }, perf@x.values, perf@y.values, pred@cutoffs)  
}  
print(opt.cut(ROCRperf, ROCRpred))

## [,1]  
## sensitivity 0.7101002  
## specificity 0.7462234  
## cutoff 0.1770680

Threshold Test Confusion Matrix

t1 = table(chicago1$Arrest,predictions > 0.1770680)  
t1

##   
## FALSE TRUE  
## FALSE 158814 54007  
## TRUE 15487 37928

Accuracy of Threshold Test

(t1[1,1]+t1[2,2])/nrow(chicago1)

## [1] 0.7389759

Model accuracy after applying suggested cutoff for threshold is 73.89759% This is worse than a predicion with no information. I recommend not using this model.

Threshold 0.12 Test

t2 = table(chicago1$Arrest,predictions > 0.12)  
t2

##   
## FALSE TRUE  
## FALSE 130284 82537  
## TRUE 10148 43267

(t2[1,1]+t1[2,2])/nrow(chicago1)

## [1] 0.6318154

As this lower threshold test shows that lowering the threshold parameter decreases the accuracy of the model.

Threshold 0.16 Test

t3 = table(chicago1$Arrest,predictions > 0.16)  
t3

##   
## FALSE TRUE  
## FALSE 150037 62784  
## TRUE 13705 39710

(t1[1,1]+t1[2,2])/nrow(chicago1)

## [1] 0.7389759

A threshold parameter of 0.16 appears to give the same accuracy (73.89759%) as the suggested parameter of0.1770680. This may be due to rounding and only minute changes occuring.

Naive Prediction

t4 = table(chicago1$Arrest,predictions > 1)   
t4

##   
## FALSE  
## FALSE 212821  
## TRUE 53415

(t4[1])/nrow(chicago1)

## [1] 0.7993697

The naive prediction has an accuracy of 79.93697%. Since the majority of incidents don’t lead to an arrest this does not tell us a lot. This is similar to the credit data we have worked with where there was a majority of cases that were not seen as delinquent. Our model is not that far off since we are using variables to predict if an Arrest was made or not. I believe that this is still a viable model for future use given the accuracy of 73.89759%.

#### Neural Network

Neural Network

# start\_time = Sys.time()  
# ctrl = trainControl(method = "cv", number = 10)  
#   
# nnetGrid <- expand.grid(size = 5, decay = 0.5)  
# set.seed(1234)  
#   
# nnetBasic = train(Arrest ~., train, method = "nnet", trControl = ctrl, verbose = FALSE)  
#   
# end\_time = Sys.time()  
# end\_time - start\_time

Save Model

# saveRDS(nnetBasic, "nnetBasic.rds")  
# saveRDS(nnetGrid, "nnetGrid.rds")

Load Model

nnetBasic = readRDS("nnetBasic.rds")  
nnetGrid = readRDS("nnetGrid.rds")

nnetBasic

## Neural Network   
##   
## 186366 samples  
## 4 predictor  
## 2 classes: 'FALSE', 'TRUE'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 167729, 167730, 167729, 167729, 167729, 167730, ...   
## Resampling results across tuning parameters:  
##   
## size decay Accuracy Kappa   
## 1 0e+00 0.8710173 0.5058881  
## 1 1e-04 0.8485185 0.3466809  
## 1 1e-01 0.8732118 0.5149865  
## 3 0e+00 0.8735928 0.5144389  
## 3 1e-04 0.8735338 0.5152091  
## 3 1e-01 0.8739952 0.5155903  
## 5 0e+00 0.8737913 0.5144905  
## 5 1e-04 0.8736626 0.5143510  
## 5 1e-01 0.8738986 0.5160264  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final values used for the model were size = 3 and decay = 0.1.

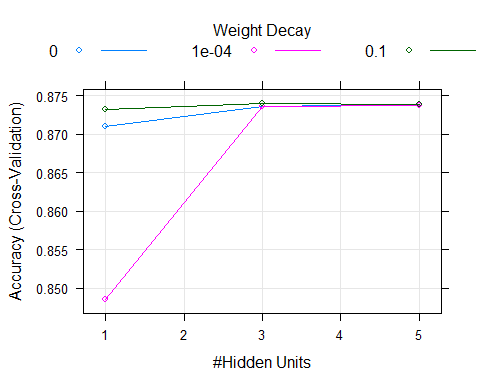
Fitted Neural Network

# start\_time = Sys.time()  
# fitctrl = trainControl(method = "cv", number = 10)  
#   
# nnetGrid <- expand.grid(size = seq(from = 2, to = 3, by = 1), decay =0.1, to = 0.2, by = .1))  
#   
# set.seed(1234)  
#   
# nnetFit = train(Arrest ~., train, method = "nnet", trControl = fitctrl, verbose = FALSE)  
#   
# end\_time = Sys.time()  
# end\_time - start\_time  
#   
# saveRDS(nnetFit, "nnetFit.rds") #These are simply here to show my process and reduce space since this model isn't working for me.  
# nnetFit = readRDS("nnetFit.rds") #These are simply here to show my process and reduce space since this model isn't working for me.

Due to spacial requirment errors I was not able to run the Fitted Neural Network. I left the code as multiple attempts were made reducing the cv value in various from 10 all the way down to 2 with errors running on every instance. I believe that this model truly enhances Neural Network predictions when it operates properly.

Plot

plot(nnetBasic)



Predictions (Train)

# predNet = predict(nnetBasic, train)

I commented this out because when I tried Knitting this it said it could not allocate vector of size 1.4 Mb. I will fully write out the results below my Confustion Matrix in case this causes an error there as well since this is commented out. In the chance the Confusion Matrix still works, I will cut the additional text in the RMD file. I am not sure why I was able to get this run just to have it error out when running it again.

Confusion Matrix (Train)

# confusionMatrix(predNet, train$Arrest, positive = "TRUE")

Confusion Matrix and Statistics

Reference

Prediction FALSE TRUE FALSE 146705 21163 TRUE 2270 16228

Accuracy : 0.8743   
 95% CI : (0.8727, 0.8758)  
No Information Rate : 0.7994   
P-Value [Acc > NIR] : < 2.2e-16   
   
 Kappa : 0.5165

Mcnemar’s Test P-Value : < 2.2e-16

Sensitivity : 0.43401   
 Specificity : 0.98476   
 Pos Pred Value : 0.87728   
 Neg Pred Value : 0.87393   
 Prevalence : 0.20063   
 Detection Rate : 0.08708

Detection Prevalence : 0.09926  
Balanced Accuracy : 0.70939

'Positive' Class : TRUE

As we can see from the confusion matrix results; accuracy of the basic model is 87.43% with a sensitivity of 43.401% and a specificity of 98.476 %. The null value accuracy is 79.94%. According to the P-value of the matrix, we have a significant increase in our prediction accuracy over the null accuracy. Sensitivity is not great but does not drag the accuracy down to an insiginficant difference point.

Predictions (Test)

predNet1 = predict(nnetBasic, test)

Confusion Matrix (Test)

confusionMatrix(predNet1, test$Arrest, positive = "TRUE")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction FALSE TRUE  
## FALSE 62837 9093  
## TRUE 1009 6931  
##   
## Accuracy : 0.8735   
## 95% CI : (0.8712, 0.8758)  
## No Information Rate : 0.7994   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.5138   
##   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Sensitivity : 0.43254   
## Specificity : 0.98420   
## Pos Pred Value : 0.87292   
## Neg Pred Value : 0.87359   
## Prevalence : 0.20063   
## Detection Rate : 0.08678   
## Detection Prevalence : 0.09941   
## Balanced Accuracy : 0.70837   
##   
## 'Positive' Class : TRUE   
##

This model responded very well (relative to the training set) to new data. I will most likely recommend this model. Accuracy, Sensitivity and Specificity all show only the slightest decrease in results.

#### Classification Trees ALL

The first set of trees will be using all of the chicago data. The second set of trees will only use my chosen predictor variables to test the difference between all the data and my assumed best predictors.

library("RColorBrewer")  
library("rattle")

## Rattle: A free graphical interface for data science with R.  
## Version 5.3.0 Copyright (c) 2006-2018 Togaware Pty Ltd.  
## Type 'rattle()' to shake, rattle, and roll your data.

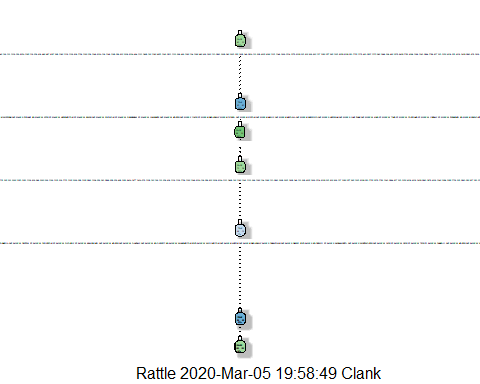
library("rpart")

set.seed(1234)  
train.rows = createDataPartition(y = chicago$Arrest, p=0.7, list = FALSE)  
train2 = chicago[train.rows,]  
test2 = chicago[-train.rows,]

Classification Tree ALL

tree1 = rpart(Arrest ~., chicago, method = "class")  
fancyRpartPlot(tree1)

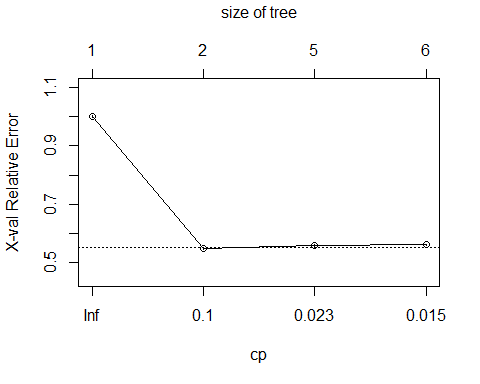
## Warning: labs do not fit even at cex 0.15, there may be some overplotting



printcp(tree1)

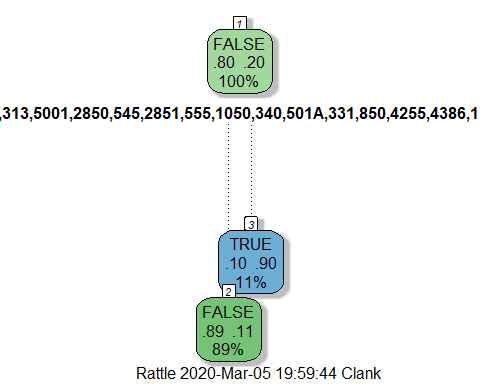
##   
## Classification tree:  
## rpart(formula = Arrest ~ ., data = chicago, method = "class")  
##   
## Variables actually used in tree construction:  
## [1] Block IUCR   
##   
## Root node error: 53415/266236 = 0.20063  
##   
## n= 266236   
##   
## CP nsplit rel error xerror xstd  
## 1 0.451596 0 1.00000 1.00000 0.0038685  
## 2 0.023090 1 0.54840 0.54917 0.0030246  
## 3 0.022035 4 0.47914 0.55825 0.0030464  
## 4 0.010000 5 0.45710 0.56093 0.0030528

plotcp(tree1)



Prune Trees

tree2 = prune(tree1, cp = tree1$cptable[which.min(tree1$cptable[,"xerror"]),"CP"])  
fancyRpartPlot(tree2)



Predictions on Training Set

treepred = predict(tree2, train2, type = "class")  
head(treepred)

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

Confusion Matrix

confusionMatrix(treepred, train2$Arrest, positive = "TRUE")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction FALSE TRUE  
## FALSE 146829 18422  
## TRUE 2146 18969  
##   
## Accuracy : 0.8896   
## 95% CI : (0.8882, 0.8911)  
## No Information Rate : 0.7994   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.5889   
##   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Sensitivity : 0.5073   
## Specificity : 0.9856   
## Pos Pred Value : 0.8984   
## Neg Pred Value : 0.8885   
## Prevalence : 0.2006   
## Detection Rate : 0.1018   
## Detection Prevalence : 0.1133   
## Balanced Accuracy : 0.7465   
##   
## 'Positive' Class : TRUE   
##

As expected, when all the data is included we get a marginally better accuracy. This shows that my predictor variables are very good since accuracy, sensitivity and specificity are only slightly better than my Neural Network. This means that the other variables would only provide a slight increase in accuracy and the expense of computational computing power may not outweigh the benefit of having the extra data.

Predictions on Testing Set

treepred\_test = predict(tree2, newdata = test2, type = "class")  
head(treepred\_test)

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

Confusion Matrix Testing Set

confusionMatrix(treepred\_test, test2$Arrest, positive = "TRUE")

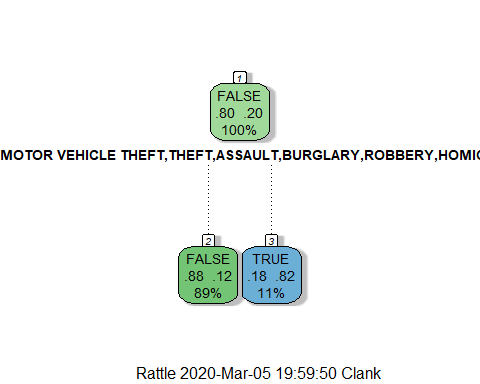
## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction FALSE TRUE  
## FALSE 62914 7793  
## TRUE 932 8231  
##   
## Accuracy : 0.8908   
## 95% CI : (0.8886, 0.8929)  
## No Information Rate : 0.7994   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.5944   
##   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Sensitivity : 0.5137   
## Specificity : 0.9854   
## Pos Pred Value : 0.8983   
## Neg Pred Value : 0.8898   
## Prevalence : 0.2006   
## Detection Rate : 0.1031   
## Detection Prevalence : 0.1147   
## Balanced Accuracy : 0.7495   
##   
## 'Positive' Class : TRUE   
##

The results here are actually better than on the training set of data. I believe that overfitting of this model is occurring when all variables are considered.

#### Classification Trees SELECTED

Classification Trees SELECTED

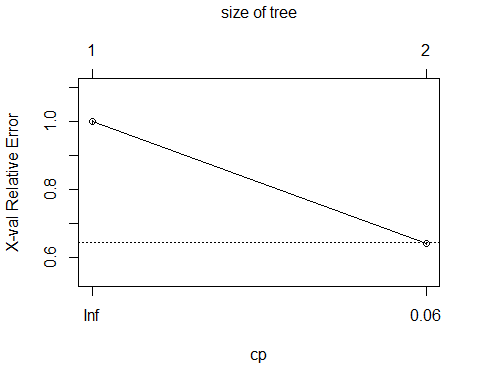
tree3 = rpart(Arrest ~., chicago1, method = "class")  
fancyRpartPlot(tree3)

 This tree is a bit concerning as it places almost all weight on the Primary Type variable. This very well may be the largest deciding factor however, as shown in other models, there is also significance in other variables.

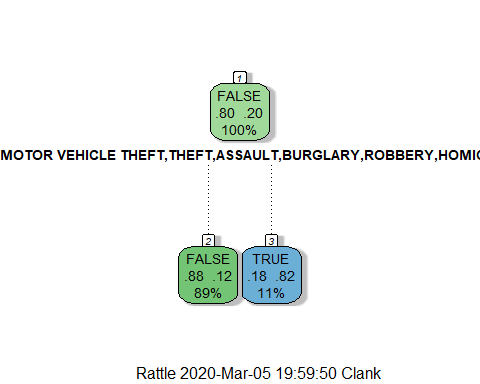
printcp(tree3)

##   
## Classification tree:  
## rpart(formula = Arrest ~ ., data = chicago1, method = "class")  
##   
## Variables actually used in tree construction:  
## [1] Primary Type  
##   
## Root node error: 53415/266236 = 0.20063  
##   
## n= 266236   
##   
## CP nsplit rel error xerror xstd  
## 1 0.3576 0 1.0000 1.00000 0.0038685  
## 2 0.0100 1 0.6424 0.64246 0.0032369

plotcp(tree3)



tree4 = prune(tree3, cp = tree1$cptable[which.min(tree1$cptable[,"xerror"]),"CP"])  
fancyRpartPlot(tree4)

 After pruning the decision tree we can see that it looks identical to the unpruned tree. This is more of a concern, seeing that other factors variables were still left out.

Predictions on Training Set

treepred2 = predict(tree4, train, type = "class")  
head(treepred)

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

Confusion Matrix

confusionMatrix(treepred2, train$Arrest, positive = "TRUE")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction FALSE TRUE  
## FALSE 145231 20264  
## TRUE 3744 17127  
##   
## Accuracy : 0.8712   
## 95% CI : (0.8696, 0.8727)  
## No Information Rate : 0.7994   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.5188   
##   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Sensitivity : 0.4581   
## Specificity : 0.9749   
## Pos Pred Value : 0.8206   
## Neg Pred Value : 0.8776   
## Prevalence : 0.2006   
## Detection Rate : 0.0919   
## Detection Prevalence : 0.1120   
## Balanced Accuracy : 0.7165   
##   
## 'Positive' Class : TRUE   
##

As we can see from the accuracy of 87.12% Other variables in other models provide only slightly more accuracy.

Predictions on Testing Set

treepred\_test2 = predict(tree4, newdata = test, type = "class")  
head(treepred\_test2)

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

Confusion Matrix Testing Set

confusionMatrix(treepred\_test2, test$Arrest, positive = "TRUE")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction FALSE TRUE  
## FALSE 62205 8665  
## TRUE 1641 7359  
##   
## Accuracy : 0.871   
## 95% CI : (0.8686, 0.8733)  
## No Information Rate : 0.7994   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.5187   
##   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Sensitivity : 0.45925   
## Specificity : 0.97430   
## Pos Pred Value : 0.81767   
## Neg Pred Value : 0.87773   
## Prevalence : 0.20063   
## Detection Rate : 0.09214   
## Detection Prevalence : 0.11268   
## Balanced Accuracy : 0.71677   
##   
## 'Positive' Class : TRUE   
##

This model performs well on new data. Accuracy is still fairly high and significantly better than the null value. The Neural Network is still marginally superior and will remain my recommended model.

#### Results

Load Results Spreadsheet

final\_results <- read\_csv("~/BAN502/Final Project/Phase\_2/final\_results.csv")

## Parsed with column specification:  
## cols(  
## Model = col\_character(),  
## Type = col\_character(),  
## Accuracy = col\_double(),  
## Null\_Accuracy = col\_double(),  
## `P-Value` = col\_character(),  
## Sensitivity = col\_double(),  
## Specificity = col\_double(),  
## Pfalse\_Rfalse = col\_double(),  
## Ptrue\_Rfalse = col\_double(),  
## Pfalse\_Rtrue = col\_double(),  
## Ptrue\_Rtrue = col\_double()  
## )

final\_results = final\_results %>% select(-"P-Value", -"Null\_Accuracy")  
show(final\_results)

## # A tibble: 9 x 9  
## Model Type Accuracy Sensitivity Specificity Pfalse\_Rfalse Ptrue\_Rfalse  
## <chr> <chr> <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 Rand~ Train 0.879 0.454 0.986 146882 2093  
## 2 Rand~ Test 0.532 0.459 0.550 35091 28755  
## 3 K-Fo~ Thre~ 0.739 0.710 0.746 158814 15487  
## 4 Neur~ Train 0.874 0.434 0.985 146705 2270  
## 5 Neur~ Test 0.874 0.433 0.984 62837 1009  
## 6 Clas~ Train 0.890 0.507 0.986 146829 2146  
## 7 Clas~ Test 0.891 0.514 0.985 62914 932  
## 8 Clas~ Train 0.871 0.458 0.975 145231 3744  
## 9 Clas~ Test 0.871 0.459 0.974 62205 1641  
## # ... with 2 more variables: Pfalse\_Rtrue <dbl>, Ptrue\_Rtrue <dbl>

I chose to make a sheet in excel in order to accurately record my results. Below I will go through some varying displays to look at my final values. For displaying the information fully, I removed P-Value as all values were <2e-16, and the Null\_Value which all had model prediction with no information values of .7994.

For the below analysis: With these results I will not include the Class. Trees ALL as they encompass all variables that I did not choose to put in my final model predictors. I ran this model to see the difference between my chosen predictors and using all variables as predictors. As we can see; class Trees ALL is only marginally better than my other predictors which reaffirms the decisions I made when choosing my predictor variables.

As we can see from the above table: The Neural Network had the best accuracy on utilizing the model in order to predict new data. Training Accuracy was 87.43% and Testing Accuracy was 87.35%.This is the model I would choose to run further tests. Unfortunately I was not able to increase accuracy by creating a Fitted Neural Network due to my computers memory restraints.

Random Forests generated good results on the Training set of data, however, when running against new data (Testing set), results were inferior. I would attempt tweaking this model to get the best results although I do not thing they would be as good as the Neural Network.

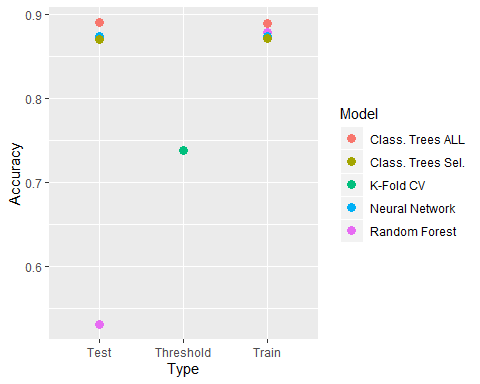
Classification Trees with the selected material also gave good results. Training Accuracy was 87.12 % and Testing Accuracy was 87.1%. This is minutely worse than the Neural Network. My biggest issue with the classification trees is that the trees seemed far too simplictic and put all decision making into 1 or 2 contstraints. With this volume of data, I would hesitate to rely so heavily on only 2 variables to predict Arrests.

K-Fold CV using a glm model was not my best decision. The results were skewed and not relevant. My results ended up being worse than the null predictor value for this data set. I would not use this model.

Below: Please note that I am still ignoring “Class. Trees ALL”. I will also exclude K-Fold CV as I do not believe it turned into a viable model.

Model Accuracy

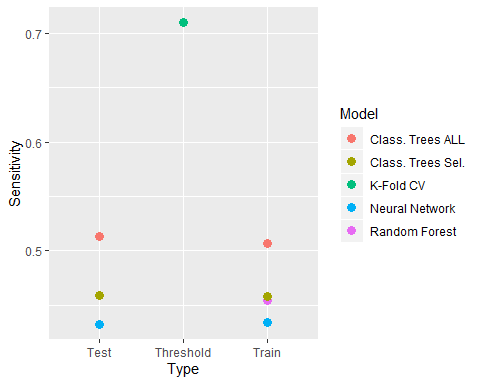
ggplot(final\_results, aes(x=Type, y = Accuracy, color = Model)) + geom\_point(size = 3)

 This graph shows that Random Forest produced the best results on the Training set of data but also the worst results when introducint new data. Neural Network is second best on Training data and best when used on Testing (new) data.

For model sensitivity and specificity it is important to note that my “positive class” was TRUE for Arrests.

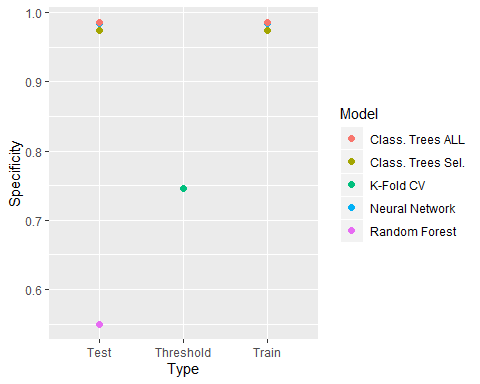
Model Sensitivity

ggplot(final\_results, aes(x=Type, y = Sensitivity, color = Model)) + geom\_point(size = 3)

 As we can see in the above graph. None of the models had a great Sensitivity (Predicting that an Arrest would occur and an Arrest actually being made). Since the majority of this data does not end in an Arrest being made it is not surprising that our model with the best Accuracy (Neural Network) also has the worst score at predicting an actual Arrest taking place. Since it is accurately guessing the majority of the data (an arrest not occuring and being correct) I believe that the weighted value of sensitivity is not great enough to drastically pull accuracy down a great extent to where the model would no longer be significant (P-value > 0.05).

Model Specficity

ggplot(final\_results, aes(x=Type, y = Specificity, color = Model)) + geom\_point(size = 2)

 As illustrated in the above graph, most of the model components scored very highly on predicing that an Arrest would not take place. since the null value accuracy of predicing an arrest is 79.94% I would say that it makes sense that the models attempting to predict this factor would be better. All models with the exception of Random Forest provide a good level of specificity on both training and testing sets of data. I would consider this information carefully since I know that most of this data does not end in an arrest.

#### Final Notes

Use Neural Network to predict new data to evaluate the chance of an Arrest being made. This data can be used for field work and possibly dispatching more officers to incidents that the model predicts would most likely end in an Arrest and dispatching fewer officers to incidents that do not. This can save departments money and lead to further risk prevention for aggressive arrest scenarios. With this data there may be a higher chance of an Arrest being made as it could lead to less incidents to disperse before officers arrived.