# Fields, Alex

## BAN 502

### Phase 2 - Final Project

#### Part 1 - Data Prep

options(tidyverse.quiet = TRUE)  
library(tidyverse)#Data Cleaning/Wrangling  
library(MASS)#Statistics  
library(caret)#ML base  
library(VIM)#Missing data  
library(lubridate)#Date and Time Functionality  
library(cluster)#algorithms for clustering  
library(factoextra)#visualization  
library(dendextend)#visualization  
library(rpart)#Decision Tree  
library(caretEnsemble) #new package  
library(ranger)#Random Forest  
library(nnet)#Nueral Network  
library(xgboost)#XGBoost  
library(RColorBrewer)#RandomForest  
library(rattle)#RandomForest  
library(GGally)#Correlation

### Import/Viewing Dataset

chicago <- read\_csv("chicago2.csv")

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

chicago = chicago[-1]#drops first column  
  
#Dataset is too big for ML Models. I will need to cut dataset in ~half.  
sample\_size = floor(0.2\*nrow(chicago))  
new\_data = sample(seq\_len(nrow(chicago)),size = sample\_size)  
chicago =chicago[new\_data,]

#### We are using only 20% of the 15000 rows in the dataset. Since this RMD file is very computationally intensive, it is needed.

### Refactoring Data

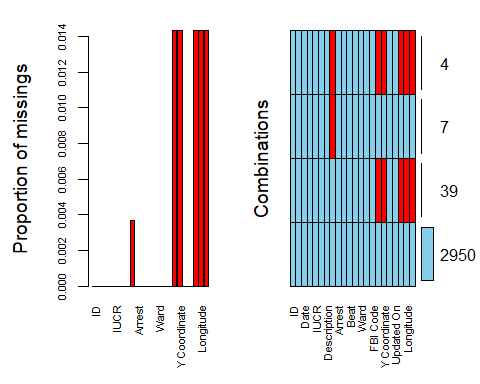
chicago = chicago %>% mutate(Date = mdy\_hms(Date))  
  
chicago = chicago %>% mutate(Arrest = as\_factor(as.character(Arrest))) %>%  
mutate(Arrest = fct\_recode(Arrest,  
"Arrested" = "TRUE",  
"Not\_Arrested" = "FALSE"))  
  
chicago = chicago %>% mutate(Domestic = as\_factor(as.character(Domestic))) %>%  
mutate(Domestic = fct\_recode(Domestic,  
"Domestic\_Violence" = "TRUE",  
"No\_Domestic\_Violence" = "FALSE"))  
  
chicago = chicago %>% mutate(`FBI Code` = as\_factor(as.character(`FBI Code`))) %>%  
mutate(`FBI Code` = fct\_recode(`FBI Code`,  
"Homicide" = "01A",  
"Sexual\_Assault" = "02",  
"Robbery" = "03",  
"Aggravated\_Assault" = "04A",  
"Agravated\_Battery" = "04B",  
"Buglary" = "05",  
"Larceny" = "06",  
"Motor\_Vehicle\_Theft" = "07",  
"Simple\_Assault" = "08A",  
"Simple\_Battery" = "08B",  
"Arson" = "09",  
"Forgery&Conterfeiting" = "10",  
"Fraud" = "11",  
#"Embezzlement" = "12",  
"Stolen\_Property" = "13",  
"Vandalism" = "14",  
"Weapons Violation" = "15",  
"Prostitution" = "16",  
"Criminal\_Sexual\_Abuse" = "17",  
"Drug Abuse" = "18",  
"Gambling" = "19",  
"Offenses\_Against\_Family" = "20",  
"Liquor\_License" = "22",  
"Disorderly\_Conduct" = "24",  
"Misc\_Offenses" = "26"))

## Warning: Unknown levels in `f`: 13

#### In the above code I am factoring Arrest, Domestic and FBI CODE to show string like data for the default data of TRUE/FALSE and integer values.

### View missing data

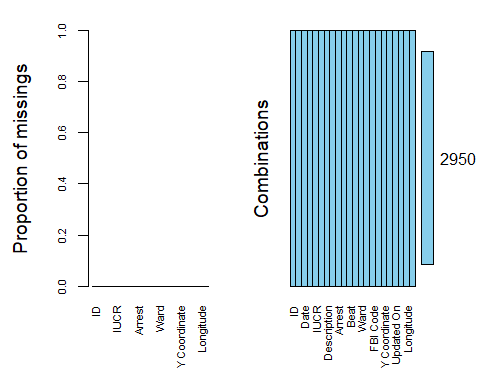
vim\_plot = aggr(chicago, numbers = TRUE, prop = c(TRUE, FALSE),cex.axis=.7)



countNA(chicago)

## [1] 226

chicago = chicago %>% drop\_na()  
vim\_plot = aggr(chicago, numbers = TRUE, prop = c(TRUE, FALSE),cex.axis=.7)



#### We are deleting all NAs in the dataset and showing what is being deleted visually.

### Data Cleansing

drop <- c("Y Coordinate","Year", "ID", "Case Number", "Updated On", "X Coordinate", "Latitude", "Longitude", "Block", "Beat", "Description", "Location Description") # Drop these variables for insignificance  
  
chicago = chicago[,!(names(chicago) %in% drop)]  
  
  
chicago = filter(chicago, `FBI Code` != 'Embezzlement' & `FBI Code` != 'Gambling' & `FBI Code` != 'Liquor License' & `FBI Code` != 'Arson' & `FBI Code` != 'Stolen Property')#Filter out unecesssary data

#### I am excluding all variables that are not needed in the dataset and filtering out certain factors that are also not important for this exercise.

## Part 2 Predicting the Data

### Training/Testing Split

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

#### We are splitting the dataset into training and testing sets.

### Logistic Regression

#logit <- glm(Arrest~Domestic+`FBI Code`+Ward+District, data = train, family="binomial")  
#summary(logit)

#saveRDS(logit,"logit.rds")  
#rm(logit)

logit = readRDS("logit.rds")  
summary(logit)

##   
## Call:  
## glm(formula = Arrest ~ Domestic + `FBI Code` + Ward + District,   
## family = "binomial", data = train)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -2.1908 -0.5368 -0.4119 -0.2378 2.6732   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -1.780e+00 3.513e-01 -5.065 4.08e-07 \*\*\*  
## DomesticDomestic\_Violence -5.950e-01 2.137e-01 -2.785 0.005356 \*\*   
## `FBI Code`Weapons Violation 2.726e+00 3.692e-01 7.383 1.54e-13 \*\*\*  
## `FBI Code`Vandalism -8.059e-01 3.739e-01 -2.155 0.031132 \*   
## `FBI Code`Simple\_Battery 8.036e-01 2.290e-01 3.509 0.000449 \*\*\*  
## `FBI Code`Misc\_Offenses 1.518e+00 2.173e-01 6.984 2.87e-12 \*\*\*  
## `FBI Code`Fraud -9.338e-01 4.810e-01 -1.941 0.052238 .   
## `FBI Code`Buglary -2.825e-01 4.244e-01 -0.665 0.505737   
## `FBI Code`Simple\_Assault 2.273e-01 3.247e-01 0.700 0.483833   
## `FBI Code`Robbery -1.011e+00 6.106e-01 -1.655 0.097860 .   
## `FBI Code`Motor\_Vehicle\_Theft -1.479e+00 7.341e-01 -2.014 0.043991 \*   
## `FBI Code`Aggravated\_Assault 4.205e-01 4.396e-01 0.957 0.338764   
## `FBI Code`Drug Abuse 1.974e+01 3.586e+02 0.055 0.956094   
## `FBI Code`Criminal\_Sexual\_Abuse 8.053e-01 6.753e-01 1.192 0.233074   
## `FBI Code`Agravated\_Battery 9.891e-01 3.486e-01 2.838 0.004544 \*\*   
## `FBI Code`Forgery&Conterfeiting 1.300e-01 7.765e-01 0.167 0.867036   
## `FBI Code`Disorderly\_Conduct 4.392e+00 7.688e-01 5.713 1.11e-08 \*\*\*  
## `FBI Code`Prostitution 2.005e+01 1.392e+03 0.014 0.988508   
## `FBI Code`Offenses\_Against\_Family -3.883e-01 1.058e+00 -0.367 0.713584   
## `FBI Code`Sexual\_Assault -1.540e+01 1.241e+03 -0.012 0.990096   
## `FBI Code`Stolen\_Property -1.537e+01 3.956e+03 -0.004 0.996901   
## Ward -4.659e-03 8.372e-03 -0.556 0.577909   
## District002 -2.723e-01 4.601e-01 -0.592 0.554017   
## District003 -9.884e-02 4.443e-01 -0.222 0.823934   
## District004 -2.610e-01 4.262e-01 -0.613 0.540194   
## District005 3.586e-02 3.991e-01 0.090 0.928413   
## District006 1.973e-01 3.851e-01 0.512 0.608323   
## District007 -3.065e-01 4.426e-01 -0.692 0.488713   
## District008 -9.924e-01 4.388e-01 -2.261 0.023730 \*   
## District009 -3.695e-01 4.760e-01 -0.776 0.437647   
## District010 -1.908e-02 4.039e-01 -0.047 0.962317   
## District011 -4.296e-01 3.974e-01 -1.081 0.279587   
## District012 -1.148e-01 4.074e-01 -0.282 0.778153   
## District014 -4.206e-01 5.045e-01 -0.834 0.404428   
## District015 -4.111e-01 4.500e-01 -0.914 0.360971   
## District016 -4.790e-01 4.849e-01 -0.988 0.323271   
## District017 -8.564e-02 4.831e-01 -0.177 0.859298   
## District018 -1.455e-01 3.934e-01 -0.370 0.711447   
## District019 -3.973e-01 5.073e-01 -0.783 0.433531   
## District020 -4.326e-01 6.116e-01 -0.707 0.479320   
## District022 1.272e-01 4.705e-01 0.270 0.786893   
## District024 1.081e-01 4.891e-01 0.221 0.825082   
## District025 -4.773e-01 4.224e-01 -1.130 0.258546   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 2065.9 on 2044 degrees of freedom  
## Residual deviance: 1370.9 on 2002 degrees of freedom  
## AIC: 1456.9  
##   
## Number of Fisher Scoring iterations: 16

#### We can see using logistic regression, using Arrest as our reponse variable and Domestic, FBI CODE, Ward and District as our predictor variables that Domestic and FBI Code are more statistically significant in predicting future arrest. We can see users who had past arrest with the following are most likely to be arrested again. Domest Violence, Weapons Violation, Simple Battery, Disorderly Conduct and Misc\_Offenses (Other).

## Random Forest

### Random Forest Generation

# fit\_control = trainControl(method = "cv",  
# number = 3) #set up 3 fold cross-validation  
#   
#   
# set.seed(1234)  
# rf\_fit = train(Arrest ~.,  
# data = train,  
# method = "ranger",  
# importance = "permutation",  
# trControl = fit\_control,  
# num.trees = 10)

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

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

## Random Forest   
##   
## 2045 samples  
## 8 predictor  
## 2 classes: 'Not\_Arrested', 'Arrested'   
##   
## No pre-processing  
## Resampling: Cross-Validated (3 fold)   
## Summary of sample sizes: 1363, 1364, 1363   
## Resampling results across tuning parameters:  
##   
## mtry splitrule Accuracy Kappa   
## 2 gini 0.7965774 0.0000000  
## 2 extratrees 0.7965774 0.0000000  
## 63 gini 0.8713933 0.5145811  
## 63 extratrees 0.8777515 0.5359545  
## 2020 gini 0.8787355 0.5592040  
## 2020 extratrees 0.8797137 0.5700416  
##   
## 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 = 2020, splitrule =  
## extratrees and min.node.size = 1.

#### We can see that running Random Forest with a 3 fold 10 tree model, that "extraTrees" was the best splitrule method showing an accuracy of 87& and a Kappa of 57%. The only downside to this is that mtrys was over 2000.

### Validating variable importance

varImp(rf\_fit)

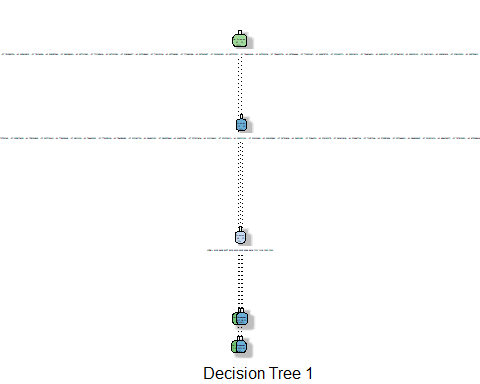
## ranger variable importance  
##   
## only 20 most important variables shown (out of 2020)  
##   
## Overall  
## `Primary Type`NARCOTICS 100.000  
## `FBI Code`Drug Abuse 41.069  
## `Primary Type`CRIMINAL TRESPASS 22.442  
## `FBI Code`Disorderly\_Conduct 18.460  
## `Primary Type`WEAPONS VIOLATION 12.979  
## `FBI Code`Weapons Violation 10.368  
## `Primary Type`PROSTITUTION 8.010  
## `FBI Code`Prostitution 5.919  
## Location(41.883500187, -87.627876698) 5.333  
## Location(41.757614433, -87.586115266) 4.505  
## Location(41.739265865, -87.604893749) 4.478  
## Location(41.903478069, -87.631433102) 3.074  
## District008 2.706  
## Location(41.899410159, -87.624131266) 2.700  
## District014 2.340  
## Location(41.851988885, -87.689219118) 1.994  
## District002 1.991  
## Location(41.88171846, -87.627760426) 1.988  
## Location(41.678519693, -87.662497935) 1.807  
## Location(41.757230688, -87.624738576) 1.805

#### Random Forest is very good in predicting variables of importance in model selection.Primary Type: Narcotics and FBI Code: Drug Abuse were the most important variables in this model. This is slightly different from what we saw with Logistic Regression.

## Building Decision Tree

tree1 = rpart(Arrest ~., train, method = "class")  
fancyRpartPlot(tree1, sub = "Decision Tree 1")

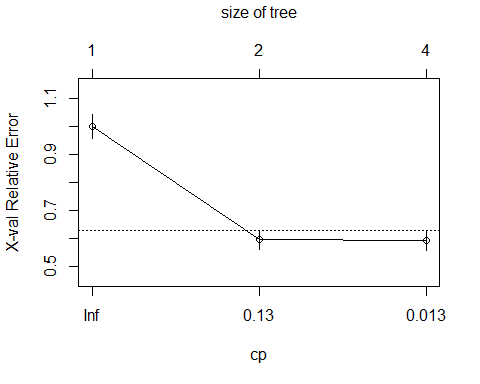
## Warning: labs do not fit even at cex 0.15, there may be some overplotting



#Showing accuracy of tree and prevents from a too complex tree  
printcp(tree1)

##   
## Classification tree:  
## rpart(formula = Arrest ~ ., data = train, method = "class")  
##   
## Variables actually used in tree construction:  
## [1] IUCR Location  
##   
## Root node error: 419/2064 = 0.203  
##   
## n= 2064   
##   
## CP nsplit rel error xerror xstd  
## 1 0.94749 0 1.000000 1.00000 0.043614  
## 2 0.01790 1 0.052506 0.59427 0.035316  
## 3 0.01000 3 0.016706 0.59189 0.035255

plotcp(tree1)

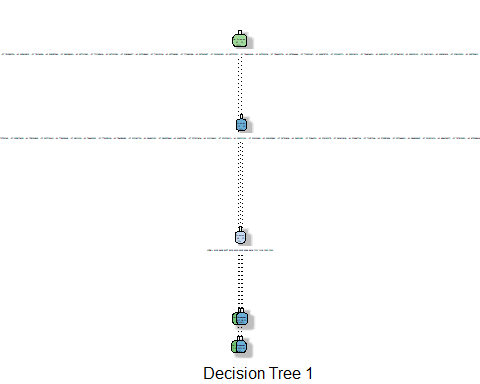


#### We can see that running Arrest as the Response variable and all others being the predictor that our Decsion Tree is is showing a node error amount of 20%. Showing an optimal Complexity Parameter of 12%. We are showing overfitting since we are showing an accuracy of close to 100%.

### CP value change

tree2 = rpart(Arrest ~., train, cp=.012, method="class")  
fancyRpartPlot(tree2, sub = "Decision Tree 1")

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



printcp(tree2)

##   
## Classification tree:  
## rpart(formula = Arrest ~ ., data = train, method = "class", cp = 0.012)  
##   
## Variables actually used in tree construction:  
## [1] IUCR Location  
##   
## Root node error: 419/2064 = 0.203  
##   
## n= 2064   
##   
## CP nsplit rel error xerror xstd  
## 1 0.94749 0 1.000000 1.00000 0.043614  
## 2 0.01790 1 0.052506 0.60859 0.035680  
## 3 0.01200 3 0.016706 0.60859 0.035680

#### This seems to be the worst of our models, the trees predictors are way too long and the tree is too accurate that we can assume overfitting. This is too good to be true even when we change the CP.

### Tree predicting

treepred = predict(tree1, train, type = "class")  
head(treepred)

## 1 2 3 4 5 6   
## Arrested Not\_Arrested Not\_Arrested Not\_Arrested Not\_Arrested Not\_Arrested   
## Levels: Not\_Arrested Arrested

### Confusion Matrix

confusionMatrix(treepred,train$Arrest,positive="Not\_Arrested") #prdictions first then actual

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction Not\_Arrested Arrested  
## Not\_Arrested 1640 2  
## Arrested 5 417  
##   
## Accuracy : 0.9966   
## 95% CI : (0.993, 0.9986)  
## No Information Rate : 0.797   
## P-Value [Acc > NIR] : <2e-16   
##   
## Kappa : 0.9895   
##   
## Mcnemar's Test P-Value : 0.4497   
##   
## Sensitivity : 0.9970   
## Specificity : 0.9952   
## Pos Pred Value : 0.9988   
## Neg Pred Value : 0.9882   
## Prevalence : 0.7970   
## Detection Rate : 0.7946   
## Detection Prevalence : 0.7955   
## Balanced Accuracy : 0.9961   
##   
## 'Positive' Class : Not\_Arrested   
##

#### Test set is showing overfitting like the train set did.

### Test Confusion Matrix

tree2 = rpart(Arrest ~., test, method = "class")  
treepred = predict(tree2, test, type = "class")  
confusionMatrix(treepred,test$Arrest,positive="Not\_Arrested") #predictions first then actual

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction Not\_Arrested Arrested  
## Not\_Arrested 703 3  
## Arrested 1 176  
##   
## Accuracy : 0.9955   
## 95% CI : (0.9884, 0.9988)  
## No Information Rate : 0.7973   
## P-Value [Acc > NIR] : <2e-16   
##   
## Kappa : 0.9859   
##   
## Mcnemar's Test P-Value : 0.6171   
##   
## Sensitivity : 0.9986   
## Specificity : 0.9832   
## Pos Pred Value : 0.9958   
## Neg Pred Value : 0.9944   
## Prevalence : 0.7973   
## Detection Rate : 0.7961   
## Detection Prevalence : 0.7995   
## Balanced Accuracy : 0.9909   
##   
## 'Positive' Class : Not\_Arrested   
##

## Nueral Network

### Building Nueral Network

drop <- c("IUCR", "Primary Type") # Drop these variables for insignificance  
  
train = train[,!(names(train) %in% drop)]  
test = test[,!(names(test) %in% drop)]  
  
#Converts all data to factors  
train[sapply(train, is.character)] <- lapply(train[sapply(train, is.character)],   
 as.factor) #Converts all chr data into factors  
  
test[sapply(test, is.character)] <- lapply(test[sapply(test, is.character)],   
 as.factor) #Converts all chr data into factors

#### We are applying all other variables as factors to run the NNET. We are also taking out IUCR since it has over 100 factors.

start\_time = Sys.time() #for timing  
fitControl = trainControl(method = "cv",  
 number = 3)  
  
set.seed(1234)  
nnetBasicTrain = train(x=as.data.frame(train[,-3]),y=train$Arrest,  
 method = "nnet",  
 #tuneGrid = nnetGrid,  
 trControl = fitControl,  
 MaxNWt = 15000,  
 trace = FALSE)  
  
end\_time = Sys.time()  
end\_time-start\_time

## Time difference of 12.83329 mins

#### We are creating out NNET with the training dataset. We are setting out Max Network Weights to over 15,000 since we have largely factored dataset.

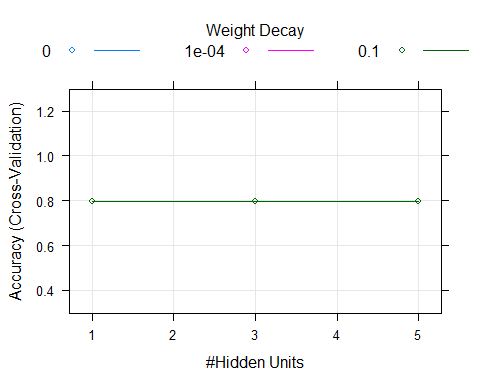
nnetBasicTrain

## Neural Network   
##   
## 2064 samples  
## 7 predictor  
## 2 classes: 'Not\_Arrested', 'Arrested'   
##   
## No pre-processing  
## Resampling: Cross-Validated (3 fold)   
## Summary of sample sizes: 1376, 1376, 1376   
## Resampling results across tuning parameters:  
##   
## size decay Accuracy Kappa  
## 1 0e+00 0.7969961 0   
## 1 1e-04 0.7969961 0   
## 1 1e-01 0.7969961 0   
## 3 0e+00 0.7969961 0   
## 3 1e-04 0.7969961 0   
## 3 1e-01 0.7969961 0   
## 5 0e+00 0.7969961 0   
## 5 1e-04 0.7969961 0   
## 5 1e-01 0.7969961 0   
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final values used for the model were size = 1 and decay = 0.1.

#### We are showing an accuracy of 79% with NNET model. THis seems relatively in line with what we would want to see. Our decay rates are constant so this could cause an issue with our model.

### Results nnet

plot(nnetBasicTrain)



predNetBasicTrain = predict(nnetBasicTrain, train)  
confusionMatrix(predNetBasicTrain, train$Arrest, positive = "Not\_Arrested")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction Not\_Arrested Arrested  
## Not\_Arrested 1645 419  
## Arrested 0 0  
##   
## Accuracy : 0.797   
## 95% CI : (0.779, 0.8142)  
## No Information Rate : 0.797   
## P-Value [Acc > NIR] : 0.5131   
##   
## Kappa : 0   
##   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 1.000   
## Specificity : 0.000   
## Pos Pred Value : 0.797   
## Neg Pred Value : NaN   
## Prevalence : 0.797   
## Detection Rate : 0.797   
## Detection Prevalence : 1.000   
## Balanced Accuracy : 0.500   
##   
## 'Positive' Class : Not\_Arrested   
##

#### Our model is showing some inaccuracies in the model. We are showing no Neg Pred values and showing the same accuracy as our Naive approach. This is not good.

### Building Nueral Network Test

start\_time = Sys.time() #for timing  
fitControl = trainControl(method = "cv",  
 number = 3)  
  
set.seed(1234)  
nnetBasicTest = train(x=as.data.frame(test[,-3]),y=test$Arrest,  
 method = "nnet",  
 #tuneGrid = nnetGrid,  
 trControl = fitControl,  
 MaxNWt = 15000,  
 trace = FALSE)  
  
end\_time = Sys.time()  
end\_time-start\_time

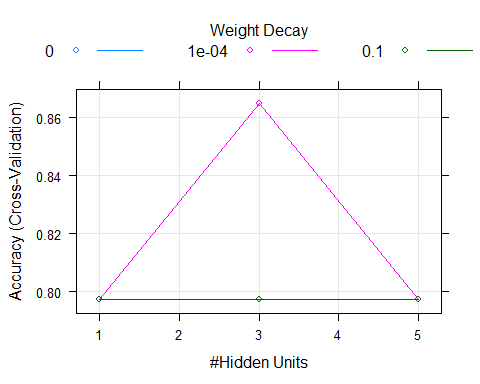
## Time difference of 1.542312 mins

nnetBasicTest

## Neural Network   
##   
## 883 samples  
## 7 predictor  
## 2 classes: 'Not\_Arrested', 'Arrested'   
##   
## No pre-processing  
## Resampling: Cross-Validated (3 fold)   
## Summary of sample sizes: 589, 589, 588   
## Resampling results across tuning parameters:  
##   
## size decay Accuracy Kappa   
## 1 0e+00 0.7972828 0.0000000  
## 1 1e-04 0.7972828 0.0000000  
## 1 1e-01 0.7972828 0.0000000  
## 3 0e+00 0.7972828 0.0000000  
## 3 1e-04 0.8650794 0.3333333  
## 3 1e-01 0.7972828 0.0000000  
## 5 0e+00 0.7972828 0.0000000  
## 5 1e-04 0.7972828 0.0000000  
## 5 1e-01 0.7972828 0.0000000  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final values used for the model were size = 3 and decay = 1e-04.

### Results nnet

plot(nnetBasicTest)



predNetBasicTest = predict(nnetBasicTest, test)  
confusionMatrix(predNetBasicTest, test$Arrest, positive = "Not\_Arrested")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction Not\_Arrested Arrested  
## Not\_Arrested 704 179  
## Arrested 0 0  
##   
## Accuracy : 0.7973   
## 95% CI : (0.7692, 0.8233)  
## No Information Rate : 0.7973   
## P-Value [Acc > NIR] : 0.52   
##   
## Kappa : 0   
##   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 1.0000   
## Specificity : 0.0000   
## Pos Pred Value : 0.7973   
## Neg Pred Value : NaN   
## Prevalence : 0.7973   
## Detection Rate : 0.7973   
## Detection Prevalence : 1.0000   
## Balanced Accuracy : 0.5000   
##   
## 'Positive' Class : Not\_Arrested   
##

#### Again, as with the Training set, the test set above is showing the same accuracy as the Naive approach. This is not the model we would want to use moving forward.

## Model Findings

### In the end looking through all models, I would suggest using Logistic Regression or Random Forest for the model to accuratly predict if someone will be arrested in the future given certain criteria.