# BAN 502 - Course Project

## Phase 2

### Runge, Laura

library(rattle)  
library(rpart)  
library(RColorBrewer)  
library(tidyverse)  
library(lubridate)  
library(VIM)  
library(gridExtra)  
library(GGally)  
library(MASS)  
library(car)  
library(ranger)  
library(mice)  
library(caret)  
library(forcats)  
library(rcompanion)  
library(nnet)  
library(caretEnsemble)  
library(xgboost)  
library(ROCR)

**CLEANING DATA**

Load Dataset

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

Select Variables of Interest  
Mutate Variables to categorized factors

chicago2=chicago %>%  
 dplyr::select(-c("ID", "Block", "Description", "IUCR","Case Number","Updated On", "X Coordinate","Y Coordinate","Location", "FBI Code","Year","X1", "Beat", "Ward", "Community Area")) %>%  
 mutate(Date = mdy\_hm(Date)) %>%  
 mutate(Hour = hour(Date)) %>%  
 mutate(DayofWeek = wday(Date)) %>%  
 mutate(Hour=as.factor(Hour))  
  
  
chicago2$`Primary Type`=as.factor(chicago2$`Primary Type`)  
chicago2$District=as.factor(chicago2$District)  
  
  
chicago2 = chicago2 %>% mutate(Arrest = as.factor(Arrest)) %>%   
 mutate(Arrest = fct\_recode(Arrest, "No" = "FALSE", "Yes" = "TRUE" ))  
  
chicago2 = chicago2 %>% mutate(Domestic = as.factor(Domestic)) %>%   
 mutate(Domestic = fct\_recode(Domestic, "No" = "FALSE", "Yes" = "TRUE" ))  
  
chicago2 = chicago2 %>% mutate(DayofWeek= as.factor(DayofWeek)) %>%   
 mutate(DayofWeek = fct\_recode(DayofWeek, "Sunday" = "1", "Monday" = "2", "Tuesday"="3", "Wednesday"="4", "Thursday"="5", "Friday"="6", "Saturday"="7" ))

Remove missing observations

chicago2=chicago2 %>%  
 dplyr::select(-c("Latitude","Longitude","Date")) %>%  
 drop\_na()

Consolidate factor levels

Location Description:

chicago2$`Location Description` = fct\_collapse(chicago2$`Location Description`,   
 "Airport" = c("AIRCRAFT","AIRPORT BUILDING NON-TERMINAL - NON-SECURE AREA","AIRPORT BUILDING NON-TERMINAL - SECURE AREA","AIRPORT EXTERIOR - NON-SECURE AREA","AIRPORT EXTERIOR - SECURE AREA","AIRPORT PARKING LOT","AIRPORT TERMINAL LOWER LEVEL - NON-SECURE AREA","AIRPORT TERMINAL LOWER LEVEL - SECURE AREA","AIRPORT TERMINAL UPPER LEVEL - NON-SECURE AREA","AIRPORT TERMINAL UPPER LEVEL - SECURE AREA","AIRPORT TRANSPORTATION SYSTEM (ATS)","AIRPORT VENDING ESTABLISHMENT"),   
 "Business/Retail/Resturant"=c("APPLIANCE STORE","ATHLETIC CLUB","AUTO / BOAT / RV DEALERSHIP","BAR OR TAVERN","BARBERSHOP","BOWLING ALLEY","CAR WASH","CLEANING STORE","COMMERCIAL / BUSINESS OFFICE","CONVENIENCE STORE","DEPARTMENT STORE","DRUG STORE","GAS STATION","GROCERY FOOD STORE","MOVIE HOUSE/THEATER","PAWN SHOP","RESTAURANT","SMALL RETAIL STORE","TAVERN/LIQUOR STORE","FACTORY/MANUFACTURING BUILDING","WAREHOUSE","ATM (AUTOMATIC TELLER MACHINE)","BANK","CURRENCY EXCHANGE","SAVINGS AND LOAN","HOTEL","HOTEL/MOTEL"),   
 "Government Building/Land"=c("FEDERAL BUILDING","FIRE STATION","GOVERNMENT BUILDING","GOVERNMENT BUILDING/PROPERTY","LIBRARY","FOREST PRESERVE","LAKEFRONT/WATERFRONT/RIVERBANK","PARK PROPERTY","JAIL / LOCK-UP FACILITY","POLICE FACILITY/VEH PARKING LOT"),   
 "Medical/Hospital"=c("ANIMAL HOSPITAL","HOSPITAL BUILDING/GROUNDS","MEDICAL/DENTAL OFFICE","NURSING HOME/RETIREMENT HOME"),  
 "Other"=c("CEMETARY","CONSTRUCTION SITE","OTHER","POOL ROOM","SPORTS ARENA/STADIUM","ABANDONED BUILDING","VACANT LOT", "VACANT LOT/LAND","CHURCH/SYNAGOGUE/PLACE OF WORSHIP"),  
 "Public Transportation"=c("CTA BUS","CTA BUS STOP","CTA GARAGE / OTHER PROPERTY","CTA PLATFORM","CTA STATION","CTA TRAIN","OTHER COMMERCIAL TRANSPORTATION","OTHER RAILROAD PROP / TRAIN DEPOT"),  
 "Residential"=c("APARTMENT","CHA APARTMENT","CHA HALLWAY/STAIRWELL/ELEVATOR","CHA PARKING LOT/GROUNDS","DRIVEWAY - RESIDENTIAL","HALLWAY","HOUSE","PORCH","RESIDENCE","RESIDENCE PORCH/HALLWAY","RESIDENCE-GARAGE","RESIDENTIAL YARD (FRONT/BACK)","YARD"),  
 "School/College"=c("COLLEGE/UNIVERSITY GROUNDS","COLLEGE/UNIVERSITY RESIDENCE HALL","DAY CARE CENTER","SCHOOL, PRIVATE, BUILDING","SCHOOL, PRIVATE, GROUNDS","SCHOOL, PUBLIC, BUILDING","SCHOOL, PUBLIC, GROUNDS"),  
 "Street"=c("ALLEY","BRIDGE","HIGHWAY/EXPRESSWAY","PARKING LOT","PARKING LOT/GARAGE(NON.RESID.)","SIDEWALK","STREET"),  
 "Vehicle/Boat"=c("TAXICAB","VEHICLE - DELIVERY TRUCK","VEHICLE - OTHER RIDE SHARE SERVICE (E.G., UBER, LYFT)","VEHICLE NON-COMMERCIAL","VEHICLE-COMMERCIAL","VEHICLE-COMMERCIAL - ENTERTAINMENT/PARTY BUS","AUTO","BOAT/WATERCRAFT"))

Primary Type:

chicago2$`Primary Type` = fct\_collapse(chicago2$`Primary Type`,  
 "Assault"="ASSAULT",  
 "Battery"="BATTERY",  
 "Burglary"="BURGLARY",  
 "Damage"="CRIMINAL DAMAGE",  
 "Disturbing the Peace"=c("PUBLIC PEACE VIOLATION","INTERFERENCE WITH PUBLIC OFFICER","OBSCENITY","INTIMIDATION"),  
 "Drugs/Alcohol"=c("NARCOTICS","LIQUOR LAW VIOLATION"),  
 "Fraud"="DECEPTIVE PRACTICE",  
 "Homicide"="HOMICIDE",  
 "Other"=c("KIDNAPPING","OTHER OFFENSE","ARSON","STALKING","OFFENSE INVOLVING CHILDREN","GAMBLING","NON-CRIMINAL"),  
 "Robbery"="ROBBERY",  
 "Sexual"=c("SEX OFFENSE","CRIM SEXUAL ASSAULT","PROSTITUTION","PUBLIC INDECENCY"),  
 "Theft"="THEFT",  
 "Auto Theft"="MOTOR VEHICLE THEFT",  
 "Trespass"="CRIMINAL TRESPASS",  
 "Weapons"=c("WEAPONS VIOLATION","CONCEALED CARRY LICENSE VIOLATION"))  
  
#summary(chicago2)  
str(chicago2)

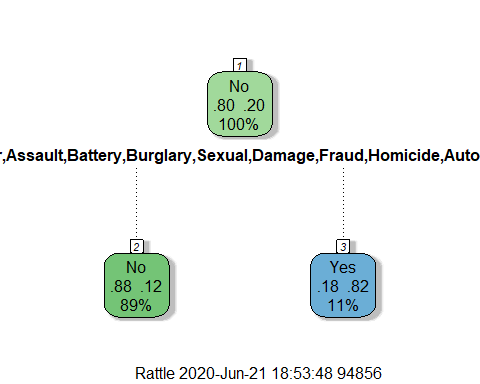
## tibble [14,944 x 7] (S3: tbl\_df/tbl/data.frame)  
## $ Primary Type : Factor w/ 15 levels "Other","Assault",..: 7 3 7 15 13 7 6 3 5 15 ...  
## $ Location Description: Factor w/ 10 levels "Other","Airport",..: 3 3 5 3 3 5 8 5 3 5 ...  
## $ Arrest : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 2 1 ...  
## $ Domestic : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 2 1 1 ...  
## $ District : Factor w/ 22 levels "1","2","3","4",..: 8 10 8 10 2 5 8 6 12 5 ...  
## $ Hour : Factor w/ 24 levels "0","1","2","3",..: 15 24 3 6 8 18 14 11 13 9 ...  
## $ DayofWeek : Factor w/ 7 levels "Sunday","Monday",..: 5 6 2 5 6 2 4 2 3 1 ...

**Training Sets**

set.seed(1234)   
train.rows = createDataPartition(y = chicago2$Arrest, p=0.7, list = FALSE)  
train = dplyr::slice(chicago2,train.rows)  
test = dplyr::slice(chicago2,-train.rows)

**Classification Tree**

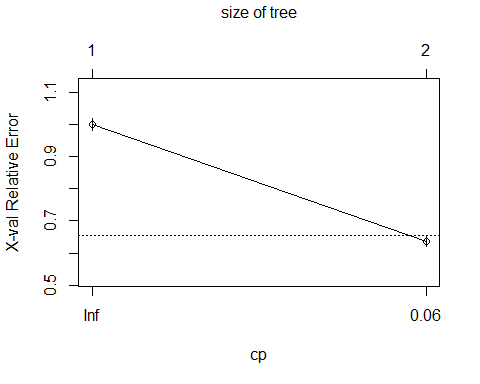
tree1 = rpart(Arrest ~., train, method="class")  
fancyRpartPlot(tree1)



printcp(tree1)

##   
## Classification tree:  
## rpart(formula = Arrest ~ ., data = train, method = "class")  
##   
## Variables actually used in tree construction:  
## [1] Primary Type  
##   
## Root node error: 2114/10461 = 0.20208  
##   
## n= 10461   
##   
## CP nsplit rel error xerror xstd  
## 1 0.36282 0 1.00000 1.00000 0.019428  
## 2 0.01000 1 0.63718 0.63718 0.016205

plotcp(tree1)



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

## 1 2 3 4 5 6   
## No No No No No No   
## Levels: No Yes

confusionMatrix(treepred,train$Arrest,positive="Yes")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 8140 1140  
## Yes 207 974  
##   
## Accuracy : 0.8712   
## 95% CI : (0.8647, 0.8776)  
## No Information Rate : 0.7979   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.5219   
##   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Sensitivity : 0.46074   
## Specificity : 0.97520   
## Pos Pred Value : 0.82472   
## Neg Pred Value : 0.87716   
## Prevalence : 0.20208   
## Detection Rate : 0.09311   
## Detection Prevalence : 0.11290   
## Balanced Accuracy : 0.71797   
##   
## 'Positive' Class : Yes   
##

treepred\_test = predict(tree1, newdata=test, type = "class")  
head(treepred\_test)

## 1 2 3 4 5 6   
## No No No No No No   
## Levels: No Yes

confusionMatrix(treepred\_test,test$Arrest,positive="Yes")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 3492 513  
## Yes 85 393  
##   
## Accuracy : 0.8666   
## 95% CI : (0.8563, 0.8764)  
## No Information Rate : 0.7979   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.4978   
##   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Sensitivity : 0.43377   
## Specificity : 0.97624   
## Pos Pred Value : 0.82218   
## Neg Pred Value : 0.87191   
## Prevalence : 0.20210   
## Detection Rate : 0.08766   
## Detection Prevalence : 0.10663   
## Balanced Accuracy : 0.70501   
##   
## 'Positive' Class : Yes   
##

**Stepwise Regression**

allmod = glm(Arrest ~., train, family = "binomial")   
#summary(allmod)

emptymod = glm(Arrest~1, train, family = "binomial")   
#summary(emptymod)

backmod = stepAIC(allmod, direction = "backward", trace = TRUE)

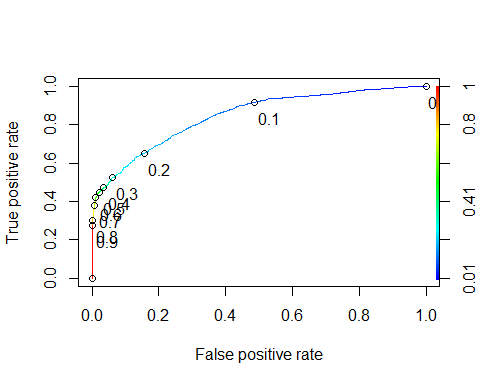
## Start: AIC=7094.99  
## Arrest ~ `Primary Type` + `Location Description` + Domestic +   
## District + Hour + DayofWeek  
##   
## Df Deviance AIC  
## - Hour 23 6967.9 7071.9  
## - DayofWeek 6 6951.0 7089.0  
## <none> 6945.0 7095.0  
## - Domestic 1 6954.0 7102.0  
## - District 21 7022.0 7130.0  
## - `Location Description` 9 7189.1 7321.1  
## - `Primary Type` 14 9770.6 9892.6  
##   
## Step: AIC=7071.94  
## Arrest ~ `Primary Type` + `Location Description` + Domestic +   
## District + DayofWeek  
##   
## Df Deviance AIC  
## - DayofWeek 6 6973.9 7065.9  
## <none> 6967.9 7071.9  
## - Domestic 1 6977.1 7079.1  
## - District 21 7045.0 7107.0  
## - `Location Description` 9 7214.8 7300.8  
## - `Primary Type` 14 9829.7 9905.7  
##   
## Step: AIC=7065.88  
## Arrest ~ `Primary Type` + `Location Description` + Domestic +   
## District  
##   
## Df Deviance AIC  
## <none> 6973.9 7065.9  
## - Domestic 1 6982.9 7072.9  
## - District 21 7051.3 7101.3  
## - `Location Description` 9 7221.9 7295.9  
## - `Primary Type` 14 9838.9 9902.9

#summary(backmod)

pred\_back = predict(backmod,train,type="response")  
head(pred\_back)

## 1 2 3 4 5 6   
## 0.05633541 0.02984200 0.06727686 0.25146867 0.28879309 0.05650493

ROCRpred = prediction(pred\_back, train$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))



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

## [1] 0.8445366

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 0.7426679  
## specificity 0.7581167  
## cutoff 0.1694621

t1 = table(train$Arrest,pred\_back > 0.1694621)  
t1

##   
## FALSE TRUE  
## No 6356 1991  
## Yes 551 1563

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

## [1] 0.7570022

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

## [1] 0.7393567

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

## [1] 0.7614712

pred\_back\_test = predict(backmod,test,type="response")  
head(pred\_back\_test)

## 1 2 3 4 5 6   
## 0.26868528 0.03261591 0.10025072 0.06418338 0.17281310 0.23320048

t2 = table(test$Arrest,pred\_back\_test > 0.1694621)  
t2

##   
## FALSE TRUE  
## No 2673 904  
## Yes 270 636

(t2[1,1]+t2[2,2])/nrow(test)

## [1] 0.7381218

(t2[2,2]/(t2[2,1]+t2[2,2]))

## [1] 0.7019868

(t2[1,1]/(t2[1,1]+t2[1,2]))

## [1] 0.7472743

**Random Forrest**

# fit\_control = trainControl(method = "cv", number = 10)  
#   
# set.seed(1234)  
# rf\_fit1 = train(x=as.matrix(train[,-3]), y=as.matrix(train$Arrest),   
# method = "ranger",   
# importance = "permutation",  
# trControl = fit\_control)

# saveRDS(rf\_fit1, "rf\_fit1.rds")  
# rm(rf\_fit1)

rf\_fit1 = readRDS("rf\_fit1.rds")  
rf\_fit1

## Random Forest   
##   
## 10461 samples  
## 6 predictor  
## 2 classes: 'No', 'Yes'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 9415, 9415, 9414, 9415, 9415, 9416, ...   
## Resampling results across tuning parameters:  
##   
## mtry splitrule Accuracy Kappa   
## 2 gini 0.8759198 0.5339020  
## 2 extratrees 0.8723843 0.4985937  
## 4 gini 0.8593830 0.5085493  
## 4 extratrees 0.8737226 0.5401234  
## 6 gini 0.8525963 0.4955929  
## 6 extratrees 0.8705681 0.5336253  
##   
## 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.

varImp(rf\_fit1)

## ranger variable importance  
##   
## Overall  
## Primary Type 100.000  
## Location Description 20.229  
## District 9.783  
## Domestic 7.285  
## Hour 2.864  
## DayofWeek 0.000

pred\_rf1 = predict.train(rf\_fit1, train)  
head(pred\_rf1)

## [1] Yes No Yes Yes No Yes  
## Levels: No Yes

confusionMatrix(pred\_rf1,train$Arrest,positive="Yes")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 4832 1770  
## Yes 3515 344  
##   
## Accuracy : 0.4948   
## 95% CI : (0.4852, 0.5044)  
## No Information Rate : 0.7979   
## P-Value [Acc > NIR] : 1   
##   
## Kappa : -0.1975   
##   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 0.16272   
## Specificity : 0.57889   
## Pos Pred Value : 0.08914   
## Neg Pred Value : 0.73190   
## Prevalence : 0.20208   
## Detection Rate : 0.03288   
## Detection Prevalence : 0.36889   
## Balanced Accuracy : 0.37081   
##   
## 'Positive' Class : Yes   
##

pred\_rf\_test1 = predict.train(rf\_fit1, test)  
head(pred\_rf\_test1)

## [1] No Yes Yes Yes No No   
## Levels: No Yes

confusionMatrix(pred\_rf\_test1, test$Arrest, positive = "Yes")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 2077 739  
## Yes 1500 167  
##   
## Accuracy : 0.5006   
## 95% CI : (0.4858, 0.5153)  
## No Information Rate : 0.7979   
## P-Value [Acc > NIR] : 1   
##   
## Kappa : -0.1789   
##   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 0.18433   
## Specificity : 0.58065   
## Pos Pred Value : 0.10018   
## Neg Pred Value : 0.73757   
## Prevalence : 0.20210   
## Detection Rate : 0.03725   
## Detection Prevalence : 0.37185   
## Balanced Accuracy : 0.38249   
##   
## 'Positive' Class : Yes   
##

**Random Forrest with mtry/Hellinger tuning**

# fit\_control = trainControl(method = "cv", number = 10)  
#   
# tunegrid = expand.grid(mtry = 1, splitrule ="hellinger", min.node.size=1)  
#   
# set.seed(1234)  
# rf\_fit = train(x=as.matrix(train[,-3]), y=as.matrix(train$Arrest),   
# method = "ranger",   
# importance = "permutation",  
# tuneGrid = tunegrid,  
# trControl = fit\_control)

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

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

## Random Forest   
##   
## 10461 samples  
## 6 predictor  
## 2 classes: 'No', 'Yes'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 9415, 9415, 9414, 9415, 9415, 9416, ...   
## Resampling results:  
##   
## Accuracy Kappa   
## 0.8177978 0.1656592  
##   
## Tuning parameter 'mtry' was held constant at a value of 1  
## Tuning  
## parameter 'splitrule' was held constant at a value of hellinger  
##   
## Tuning parameter 'min.node.size' was held constant at a value of 1

varImp(rf\_fit)

## ranger variable importance  
##   
## Overall  
## Primary Type 100.000  
## Location Description 20.839  
## District 18.143  
## Domestic 9.176  
## Hour 4.243  
## DayofWeek 0.000

pred\_rf = predict.train(rf\_fit, train)  
head(pred\_rf)

## [1] No No Yes Yes No No   
## Levels: No Yes

confusionMatrix(pred\_rf,train$Arrest,positive="Yes")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 6703 1942  
## Yes 1644 172  
##   
## Accuracy : 0.6572   
## 95% CI : (0.648, 0.6663)  
## No Information Rate : 0.7979   
## P-Value [Acc > NIR] : 1   
##   
## Kappa : -0.122   
##   
## Mcnemar's Test P-Value : 7.062e-07   
##   
## Sensitivity : 0.08136   
## Specificity : 0.80304   
## Pos Pred Value : 0.09471   
## Neg Pred Value : 0.77536   
## Prevalence : 0.20208   
## Detection Rate : 0.01644   
## Detection Prevalence : 0.17360   
## Balanced Accuracy : 0.44220   
##   
## 'Positive' Class : Yes   
##

pred\_rf\_test = predict.train(rf\_fit, test)  
head(pred\_rf\_test)

## [1] No No Yes No No No   
## Levels: No Yes

confusionMatrix(pred\_rf\_test, test$Arrest, positive = "Yes")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 2864 806  
## Yes 713 100  
##   
## Accuracy : 0.6612   
## 95% CI : (0.6471, 0.675)  
## No Information Rate : 0.7979   
## P-Value [Acc > NIR] : 1.00000   
##   
## Kappa : -0.0925   
##   
## Mcnemar's Test P-Value : 0.01825   
##   
## Sensitivity : 0.11038   
## Specificity : 0.80067   
## Pos Pred Value : 0.12300   
## Neg Pred Value : 0.78038   
## Prevalence : 0.20210   
## Detection Rate : 0.02231   
## Detection Prevalence : 0.18135   
## Balanced Accuracy : 0.45552   
##   
## 'Positive' Class : Yes   
##

**Neural Network**

# fitControl = trainControl(method = "cv", number = 5)  
#   
# nnetGrid = expand.grid(size = 1:13, decay = c(0.5, 0.1, 1e-2, 1e-3, 1e-4, 1e-5, 1e-6, 1e-7))  
#   
# set.seed(1234)  
# nnetFit = train(x=train[,-3],y=train$Arrest,   
# method = "nnet",  
# trControl = fitControl,  
# tuneGrid = nnetGrid,  
# verbose = FALSE,  
# trace = FALSE)

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

nnetFit = readRDS("nnetFit.rds")  
#nnetFit

prednnet = predict(nnetFit, train)  
head(prednnet)

## [1] No No No Yes No No   
## Levels: No Yes

confusionMatrix(prednnet, train$Arrest, positive = "Yes")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 8196 1044  
## Yes 151 1070  
##   
## Accuracy : 0.8858   
## 95% CI : (0.8795, 0.8918)  
## No Information Rate : 0.7979   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.5794   
##   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Sensitivity : 0.5061   
## Specificity : 0.9819   
## Pos Pred Value : 0.8763   
## Neg Pred Value : 0.8870   
## Prevalence : 0.2021   
## Detection Rate : 0.1023   
## Detection Prevalence : 0.1167   
## Balanced Accuracy : 0.7440   
##   
## 'Positive' Class : Yes   
##

prednnet\_test = predict(nnetFit, test)  
head(prednnet\_test)

## [1] No No No No No No  
## Levels: No Yes

confusionMatrix(prednnet\_test, test$Arrest, positive = "Yes")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 3491 504  
## Yes 86 402  
##   
## Accuracy : 0.8684   
## 95% CI : (0.8581, 0.8782)  
## No Information Rate : 0.7979   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.507   
##   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Sensitivity : 0.44371   
## Specificity : 0.97596   
## Pos Pred Value : 0.82377   
## Neg Pred Value : 0.87384   
## Prevalence : 0.20210   
## Detection Rate : 0.08967   
## Detection Prevalence : 0.10886   
## Balanced Accuracy : 0.70983   
##   
## 'Positive' Class : Yes   
##

**Ensemble**

control = trainControl(  
 method = "cv",  
 number = 5,  
 savePredictions = "final",  
 classProbs = TRUE,  
 summaryFunction = twoClassSummary,  
 index=createResample(train$Arrest))

# set.seed(1234)  
# model\_list = caretList(x=as.data.frame(train[,-3]), y=train$Arrest,  
# metric = "ROC",  
# trControl= control,  
# methodList=c("glm","rpart","ranger"),  
# tuneList=list(  
# nn = caretModelSpec(method="nnet", tuneGrid =  
# expand.grid(size = 1:80,  
# decay = c(0.5, 0.1, 1e-2, 1e-3, 1e-4, 1e-5, 1e-6, 1e-7)),trace=FALSE)))

# saveRDS(model\_list,"model\_list.rds")  
# rm(model\_list)

model\_list = readRDS("model\_list.rds")

as.data.frame(predict(model\_list, newdata=head(train)))

## nn glm rpart ranger  
## 1 0.9760825 0.9443595 0.8771552 0.9413627  
## 2 0.9819389 0.9717685 0.8771552 0.9470743  
## 3 0.9906043 0.9359571 0.8771552 0.9616719  
## 4 0.7225115 0.7503211 0.8771552 0.5214331  
## 5 0.7682052 0.7313821 0.8771552 0.8674663  
## 6 0.9829898 0.9350661 0.8771552 0.9710906

modelCor(resamples(model\_list))

## nn glm rpart ranger  
## nn 1.0000000 0.87218742 0.13864793 0.7044727  
## glm 0.8721874 1.00000000 0.07569037 0.8220869  
## rpart 0.1386479 0.07569037 1.00000000 0.3349135  
## ranger 0.7044727 0.82208687 0.33491350 1.0000000

ensemble = caretEnsemble(  
 model\_list,   
 metric="ROC",  
 trControl=control)  
  
summary(ensemble)

## The following models were ensembled: nn, glm, rpart, ranger   
## They were weighted:   
## 4.6923 -2.39 -0.7601 -0.3447 -4.5011  
## The resulting ROC is: 0.856  
## The fit for each individual model on the ROC is:   
## method ROC ROCSD  
## nn 0.8518811 0.007737934  
## glm 0.8345254 0.007460516  
## rpart 0.7561261 0.046986809  
## ranger 0.8460511 0.006193759

pred\_ensemble = predict(ensemble, train, type = "raw")  
confusionMatrix(pred\_ensemble,train$Arrest, positive = "Yes")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 8270 894  
## Yes 77 1220  
##   
## Accuracy : 0.9072   
## 95% CI : (0.9015, 0.9127)  
## No Information Rate : 0.7979   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.6636   
##   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Sensitivity : 0.5771   
## Specificity : 0.9908   
## Pos Pred Value : 0.9406   
## Neg Pred Value : 0.9024   
## Prevalence : 0.2021   
## Detection Rate : 0.1166   
## Detection Prevalence : 0.1240   
## Balanced Accuracy : 0.7839   
##   
## 'Positive' Class : Yes   
##

pred\_ensemble\_test = predict(ensemble, test, type = "raw")  
confusionMatrix(pred\_ensemble\_test,test$Arrest, positive = "Yes")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 3503 504  
## Yes 74 402  
##   
## Accuracy : 0.8711   
## 95% CI : (0.8609, 0.8807)  
## No Information Rate : 0.7979   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.5141   
##   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Sensitivity : 0.44371   
## Specificity : 0.97931   
## Pos Pred Value : 0.84454   
## Neg Pred Value : 0.87422   
## Prevalence : 0.20210   
## Detection Rate : 0.08967   
## Detection Prevalence : 0.10618   
## Balanced Accuracy : 0.71151   
##   
## 'Positive' Class : Yes   
##

**Stacked**

control2 = trainControl(  
 method = "cv",  
 number = 10,  
 savePredictions = "final",  
 classProbs = TRUE,  
 summaryFunction = twoClassSummary,  
 index=createResample(train$Arrest))  
  
stack = caretStack(  
 model\_list,  
 method ="glm",  
 metric ="ROC",  
 trControl = control2)

print(stack)

## A glm ensemble of 4 base models: nn, glm, rpart, ranger  
##   
## Ensemble results:  
## Generalized Linear Model   
##   
## 38585 samples  
## 4 predictor  
## 2 classes: 'No', 'Yes'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 10461, 10461, 10461, 10461, 10461, 10461, ...   
## Resampling results:  
##   
## ROC Sens Spec   
## 0.8564285 0.9734534 0.4997379

summary(stack)

##   
## Call:  
## NULL  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -2.6656 -0.5014 -0.3644 -0.3067 2.5279   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) 4.69234 0.09296 50.475 < 2e-16 \*\*\*  
## nn -2.38998 0.15374 -15.545 < 2e-16 \*\*\*  
## glm -0.76007 0.16759 -4.535 5.75e-06 \*\*\*  
## rpart -0.34470 0.13410 -2.570 0.0102 \*   
## ranger -4.50114 0.19851 -22.674 < 2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 38960 on 38584 degrees of freedom  
## Residual deviance: 25184 on 38580 degrees of freedom  
## AIC: 25194  
##   
## Number of Fisher Scoring iterations: 5

pred\_stack = predict(stack, train, type = "raw")  
confusionMatrix(pred\_stack,train$Arrest, positive = "Yes")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 8270 894  
## Yes 77 1220  
##   
## Accuracy : 0.9072   
## 95% CI : (0.9015, 0.9127)  
## No Information Rate : 0.7979   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.6636   
##   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Sensitivity : 0.5771   
## Specificity : 0.9908   
## Pos Pred Value : 0.9406   
## Neg Pred Value : 0.9024   
## Prevalence : 0.2021   
## Detection Rate : 0.1166   
## Detection Prevalence : 0.1240   
## Balanced Accuracy : 0.7839   
##   
## 'Positive' Class : Yes   
##

pred\_stack\_test = predict(stack, test, type = "raw")  
confusionMatrix(pred\_stack\_test,test$Arrest, positive = "Yes")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 3503 504  
## Yes 74 402  
##   
## Accuracy : 0.8711   
## 95% CI : (0.8609, 0.8807)  
## No Information Rate : 0.7979   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.5141   
##   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Sensitivity : 0.44371   
## Specificity : 0.97931   
## Pos Pred Value : 0.84454   
## Neg Pred Value : 0.87422   
## Prevalence : 0.20210   
## Detection Rate : 0.08967   
## Detection Prevalence : 0.10618   
## Balanced Accuracy : 0.71151   
##   
## 'Positive' Class : Yes   
##

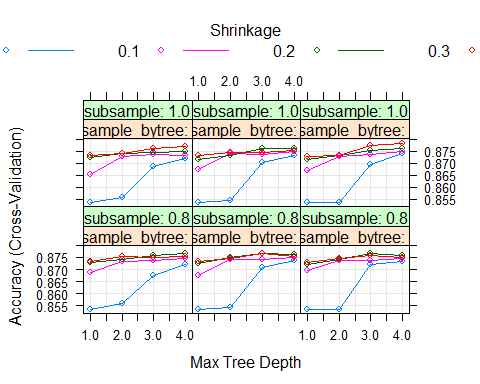
**XgBoost**

train\_dummy = dummyVars(" ~ .", data = train)  
train\_xgb = data.frame(predict(train\_dummy, newdata = train))  
#str(train\_xgb)  
  
test\_dummy = dummyVars(" ~ .", data = test)  
test\_xgb = data.frame(predict(test\_dummy, newdata = test))  
  
train\_xgb = train\_xgb %>% dplyr::select(-Arrest.No)   
test\_xgb = test\_xgb %>% dplyr::select(-Arrest.No)  
  
#str(train\_xgb)  
#str(test\_xgb)

# set.seed(1234)  
# ctrl = trainControl(method = "cv",number = 5)  
#   
# tgrid = expand.grid(  
# nrounds = 100,  
# max\_depth = c(1,2,3,4),  
# eta = c(0.01, 0.1, 0.2, 0.3),  
# gamma = 0,  
# colsample\_bytree = c(0.6, 0.8, 1),  
# min\_child\_weight = 1,  
# subsample = c(0.8, 1))  
#   
# fitxgb = train(as.factor(Arrest.Yes)~.,  
# data = train\_xgb,  
# method="xgbTree",  
# tuneGrid = tgrid,  
# trControl=ctrl)

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

fitxgb = readRDS("fitxgb.rds")  
  
#fitxgb  
plot(fitxgb)



predxgbtrain = predict(fitxgb, train\_xgb)  
confusionMatrix(as.factor(train\_xgb$Arrest.Yes), predxgbtrain,positive="1")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 8248 99  
## 1 1044 1070  
##   
## Accuracy : 0.8907   
## 95% CI : (0.8846, 0.8967)  
## No Information Rate : 0.8883   
## P-Value [Acc > NIR] : 0.2148   
##   
## Kappa : 0.5933   
##   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 0.9153   
## Specificity : 0.8876   
## Pos Pred Value : 0.5061   
## Neg Pred Value : 0.9881   
## Prevalence : 0.1117   
## Detection Rate : 0.1023   
## Detection Prevalence : 0.2021   
## Balanced Accuracy : 0.9015   
##   
## 'Positive' Class : 1   
##

predxgbtest = predict(fitxgb, test\_xgb)  
confusionMatrix(as.factor(test\_xgb$Arrest.Yes), predxgbtest,positive="1")

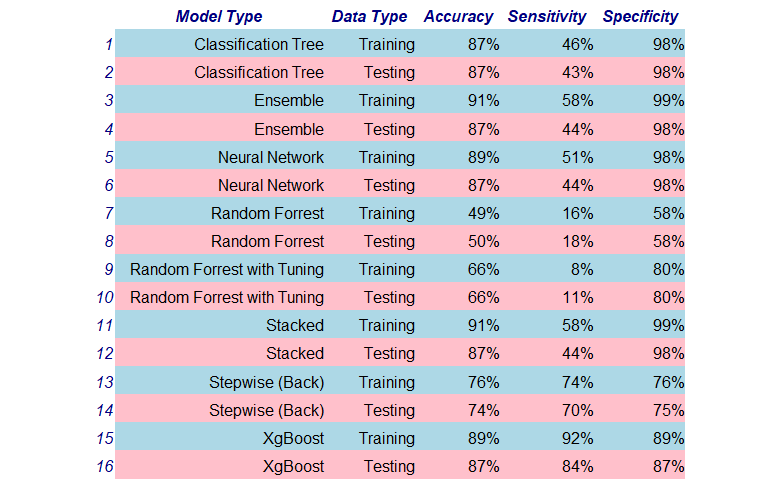
## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 3504 73  
## 1 515 391  
##   
## Accuracy : 0.8688   
## 95% CI : (0.8586, 0.8786)  
## No Information Rate : 0.8965   
## P-Value [Acc > NIR] : 1   
##   
## Kappa : 0.5027   
##   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 0.84267   
## Specificity : 0.87186   
## Pos Pred Value : 0.43157   
## Neg Pred Value : 0.97959   
## Prevalence : 0.10350   
## Detection Rate : 0.08722   
## Detection Prevalence : 0.20210   
## Balanced Accuracy : 0.85727   
##   
## 'Positive' Class : 1   
##

**Comparing Models**

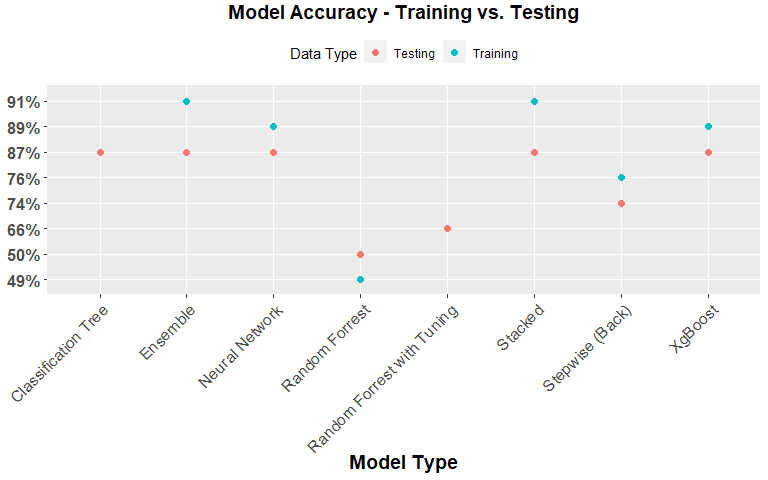
ModelComparisons2 <- read\_csv("ModelComparisons2.csv")

ModelComparisons2=ModelComparisons2 %>% drop\_na()  
  
ModelComparisons2 = ModelComparisons2 %>%  
 mutate(`Model Type`=as.factor(`Model Type`))%>%  
 mutate(`Data Type`=as.factor(`Data Type`))  
  
#ModelComparisons2

tt3 <- ttheme\_minimal(  
 core = list(fg\_params=list(hjust = 1, x=1),  
 bg\_params=list(fill=c("lightblue", "pink"))),  
 colhead=list(fg\_params=list(col="navyblue", fontface=4L)),  
 rowhead=list(fg\_params=list(col="navyblue", fontface=3L)))  
  
grid.arrange(  
 tableGrob(ModelComparisons2[1:5], theme=tt3),  
 nrow=1)



p1=ggplot(ModelComparisons2,aes(x=`Model Type`,y=Accuracy,color=`Data Type`))+  
 geom\_point(size=2)+  
 ggtitle("Model Accuracy - Training vs. Testing")+  
 theme(axis.title.y = element\_blank(),  
 axis.text.x = element\_text(size=12,angle = 45, hjust = 1),  
 axis.title.x = element\_text(size = 14, face = "bold"),  
 axis.text.y = element\_text(size=12, face= "bold"),  
 plot.title=element\_text( size=14, hjust=.5, vjust=2, face='bold'),  
 legend.position = "top")  
p1

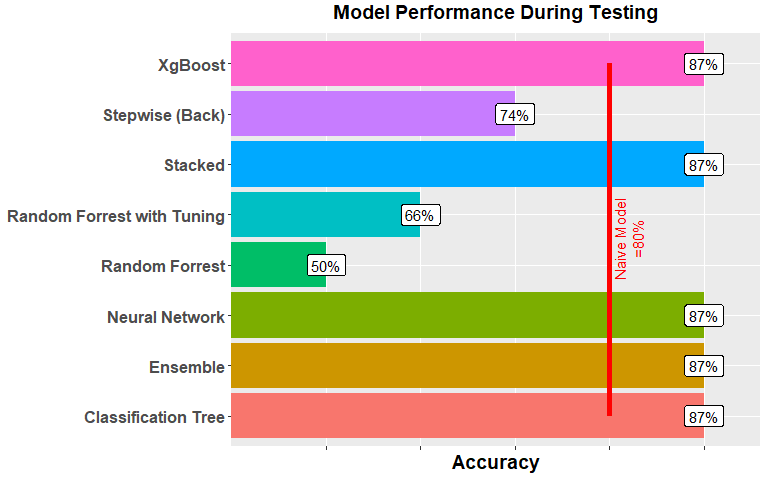


ModelComparisons <- read\_csv("ModelComparisons.csv")

ModelComparisons1=ModelComparisons %>% drop\_na()  
  
ModelComparisons1 = ModelComparisons1 %>%  
 mutate(`Model Type`=as.factor(`Model Type`))  
  
#ModelComparisons1

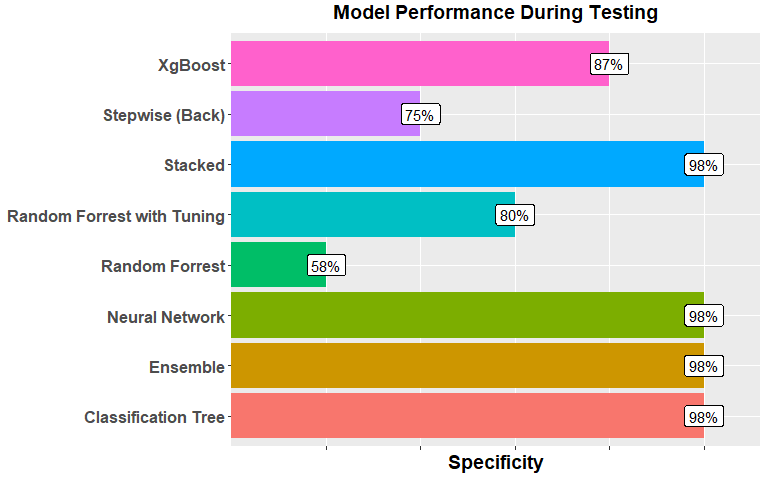
**Accuracy**

p2=ggplot(ModelComparisons1,aes(x=`Model Type`,y=Accuracy))+   
 geom\_bar(position = 'dodge',stat = "identity",aes(fill = `Model Type`))+  
 geom\_label(aes(label=Accuracy))+  
 geom\_line(aes(y = `Naive Model`), size = 2, color="red", group = 1)+  
 ggtitle("Model Performance During Testing")+  
 theme(axis.title.y = element\_blank(),  
 axis.text.x = element\_blank(),  
 axis.title.x = element\_text(size = 14, face = "bold"),  
 axis.text.y = element\_text(size=12, face= "bold"),  
 plot.title=element\_text( size=14, hjust=.5, vjust=2, face='bold'),  
 legend.position = "none")+  
 annotate('text',x=4.5,y=4.1,label="Naive Model",color="red",size=4,angle=90)+  
 annotate('text',x=4.5,y=4.3,label="=80%",color="red",size=4,angle=90)+  
 coord\_flip()  
p2



**Specificity**

p3=ggplot(ModelComparisons1,aes(x=`Model Type`,y=Specificity))+  
 geom\_bar(position = 'dodge',stat = "identity",aes(fill = `Model Type`))+  
 geom\_label(aes(label=Specificity))+  
 ggtitle("Model Performance During Testing")+  
 theme(axis.title.y = element\_blank(),  
 axis.text.x = element\_blank(),  
 axis.title.x = element\_text(size = 14, face = "bold"),  
 axis.text.y = element\_text(size=12, face= "bold"),  
 plot.title=element\_text( size=14, hjust=.5, vjust=2, face='bold'),  
 legend.position = "none")+  
 coord\_flip()  
p3



**Sensitivity**

p4=ggplot(ModelComparisons1,aes(x=`Model Type`,y=Sensitivity))+  
 geom\_bar(position = 'dodge',stat = "identity",aes(fill = `Model Type`))+  
 geom\_label(aes(label=Sensitivity))+  
 ggtitle("Model Performance During Testing")+  
 theme(axis.title.y = element\_blank(),  
 axis.text.x = element\_blank(),  
 axis.title.x = element\_text(size = 14, face = "bold"),  
 axis.text.y = element\_text(size=12, face= "bold"),  
 plot.title=element\_text( size=14, hjust=.5, vjust=2, face='bold'),  
 legend.position = "none")+  
 coord\_flip()  
p4

