# Fields, Alex

## BAN 502 - Module 5

### Assignment 1 - Finance

### Dataset Link - <https://www.kaggle.com/cnic92/200-financial-indicators-of-us-stocks-20142018>

This Dataset is from a Binary Classification competition held by Kaggle to predict stock prices

#### Observations/Evaluations

1. When looking into the “trained” Nueral Network, we can see that there is a ~72% accuracy of overall prediction. This is our first sign of a good model. This overall prediction is higer than the Naive approach (No Info Rate) which is our next sign of a good model. We see a low p-value along with a kappa score of 33%. For our dataset not being perfectly balanced, I am happy with these results.
2. When viewing our test set, we can see that their is very little change in comparison to training. Accuracy remains at 72% while Naive is below overlal accuracy and Kappa is ~31%. Both models seem fairly good to me.
3. Looking at the ensemble method, we can see utilizing *Random Forest, Decision tree, Logistic Regression and Nueral Networks* that we are not showing any substantial correlation (+-). This is what we would ideally like to see. The closest correlation we are seeing is between logistic and nnet.
4. With XGBoost, we had to convert all categorical data to Dummy variables. When running the model, I found that this was the least helpful in determinig any predictions. The accuracy of this model was below what the Naive approach was showing. As well, our plot showing tree depth was also inconclusive of anything we would want to see.
5. Since I have run a Nueral Network, An ensemble (stacked and not stacked) and XGBoost, it seems the most accurate model was the ensemble model.

### Librarying in Packages

### Importing Data

xFin <- read\_csv("2018Fin.csv")#reads in data

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

#str(xFin)  
#summary(xFin)

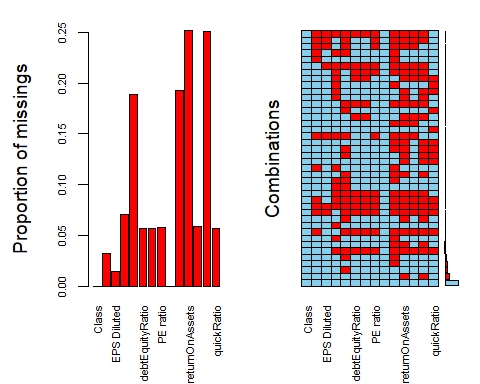
### Data Cleaning 1.1

xFin = as.data.frame(xFin)  
  
xFin = xFin %>% dplyr::select("Class",  
`Revenue Growth`, `EPS Diluted`, `EBITDA Margin`, "priceBookValueRatio", "debtEquityRatio", "debtRatio", `PE ratio`, "Sector", `5Y Revenue Growth (per Share)`, "returnOnAssets", "returnOnEquity", "returnOnCapitalEmployed",  
"quickRatio")#removing variables of unimportance  
  
#Factor Class and Sector Variables for future feature selection  
xFin = xFin %>%   
 mutate(Class = as.factor(Class)) %>%   
 mutate(Class = fct\_recode(Class, "No" = "0", "Yes" = "1" )) %>%  
 mutate(Sector = as.factor(Sector))

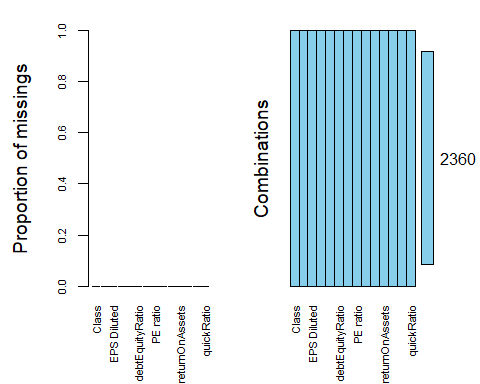
### Data Cleaning 1.2 (Missingness)

vim\_plot = aggr(xFin, numbers = TRUE, prop = c(TRUE, FALSE),cex.axis=.7)#total rows before droping NA's = 4392

## Warning in plot.aggr(res, ...): not enough vertical space to display frequencies  
## (too many combinations)

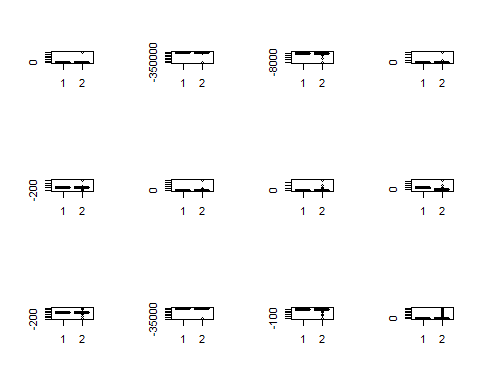


xFin = drop\_na(xFin)  
  
vim\_plot = aggr(xFin, numbers = TRUE, prop = c(TRUE, FALSE),cex.axis=.7)#total rows after droping NA's = 2630



### Testing for Outliers

par(mfrow=c(3,4))  
boxplot(xFin$Class, xFin$`Revenue Growth`)  
boxplot(xFin$Class, xFin$`EPS Diluted`)  
boxplot(xFin$Class, xFin$`EBITDA Margin`)  
boxplot(xFin$Class, xFin$priceBookValueRatio)  
boxplot(xFin$Class, xFin$debtEquityRatio)  
boxplot(xFin$Class, xFin$debtRatio)  
boxplot(xFin$Class, xFin$`PE ratio`)  
boxplot(xFin$Class, xFin$`5Y Revenue Growth (per Share)`)  
boxplot(xFin$Class, xFin$returnOnAssets)  
boxplot(xFin$Class, xFin$returnOnEquity)  
boxplot(xFin$Class, xFin$returnOnCapitalEmployed)  
boxplot(xFin$Class, xFin$quickRatio)



### Filtering Outliers

xFin = xFin %>% filter(`Revenue Growth` <= 1)  
xFin = xFin %>% filter(`EPS Diluted` >= -10, `EPS Diluted` <= 10)  
xFin = xFin %>% filter(`EBITDA Margin` >= -5, `EBITDA Margin` <= 5)  
xFin = xFin %>% filter(priceBookValueRatio >= 0, priceBookValueRatio <= 5)  
xFin = xFin %>% filter(debtEquityRatio >= -1, debtEquityRatio <= 2)  
xFin = xFin %>% filter(debtRatio <= 1)  
xFin = xFin %>% filter(`PE ratio` <= 100)  
xFin = xFin %>% filter(returnOnAssets >= -5, returnOnAssets <= 5)  
xFin = xFin %>% filter(returnOnEquity >= -5, returnOnEquity <= 5)  
xFin = xFin %>% filter(returnOnCapitalEmployed >= -2, returnOnCapitalEmployed <= 2)  
xFin = xFin %>% filter(quickRatio <= 20)

### Training/Test Split

set.seed(12345)  
train.rows = createDataPartition(xFin$Class,p=0.7,list=FALSE)  
train = dplyr::slice(xFin,train.rows)  
test = dplyr::slice(xFin,-train.rows)

### Building Nueral Network with no tuning

start\_time = Sys.time() #for timing  
fitControl = trainControl(method = "cv",   
 number = 10)  
  
nnetGrid = expand.grid(size = 1:13,  
 decay = c(0.5, 0.1, 1e-2, 1e-3, 1e-4, 1e-5, 1e-6, 1e-7))  
  
  
#xFin[,-1] removes first column from dataset, we will wawnt to exclude "Class" since this is our response variable  
set.seed(1234)  
nnetFit = train(x=xFin[,-1],y=xFin$Class,   
 method = "nnet",  
 trControl = fitControl,  
 tuneGrid = nnetGrid,  
 verbose = FALSE,  
 trace = FALSE)  
  
end\_time = Sys.time()  
end\_time-start\_time

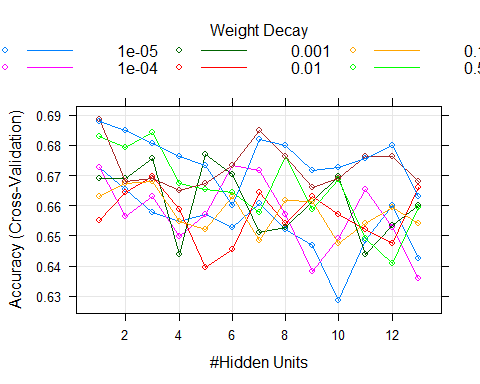
## Time difference of 6.815348 mins

#~ 10 min to run on 1300 rows and 14 columns

### Viewing NN output

#nnetFit

plot(nnetFit)



### Predictions on the NN training set

predNetFit = predict(nnetFit, train)

### Confusion matrix (Train)

confusionMatrix(predNetFit, train$Class, positive = "Yes")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 138 95  
## Yes 183 538  
##   
## Accuracy : 0.7086   
## 95% CI : (0.6786, 0.7373)  
## No Information Rate : 0.6635   
## P-Value [Acc > NIR] : 0.001625   
##   
## Kappa : 0.3001   
##   
## Mcnemar's Test P-Value : 1.81e-07   
##   
## Sensitivity : 0.8499   
## Specificity : 0.4299   
## Pos Pred Value : 0.7462   
## Neg Pred Value : 0.5923   
## Prevalence : 0.6635   
## Detection Rate : 0.5639   
## Detection Prevalence : 0.7558   
## Balanced Accuracy : 0.6399   
##   
## 'Positive' Class : Yes   
##

### Predictions on the NN Testing set

predNetFit = predict(nnetFit, test)

### Confusion matrix (Test)

confusionMatrix(predNetFit, test$Class, positive = "Yes")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 47 45  
## Yes 90 226  
##   
## Accuracy : 0.6691   
## 95% CI : (0.6211, 0.7146)  
## No Information Rate : 0.6642   
## P-Value [Acc > NIR] : 0.4397493   
##   
## Kappa : 0.1927   
##   
## Mcnemar's Test P-Value : 0.0001525   
##   
## Sensitivity : 0.8339   
## Specificity : 0.3431   
## Pos Pred Value : 0.7152   
## Neg Pred Value : 0.5109   
## Prevalence : 0.6642   
## Detection Rate : 0.5539   
## Detection Prevalence : 0.7745   
## Balanced Accuracy : 0.5885   
##   
## 'Positive' Class : Yes   
##

## Ensemble Methods

### This step builds the models in the list.

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

## glm rpart ranger nnet  
## 1 0.22056786 0.2286213 0.056 0.18882451  
## 2 0.10481112 0.2286213 0.028 0.07486191  
## 3 0.08503933 0.2286213 0.156 0.45337634  
## 4 0.57672520 0.2286213 0.124 0.18433371  
## 5 0.10852095 0.2286213 0.114 0.05893575  
## 6 0.21416066 0.2286213 0.050 0.09834058

### Model Correlation

modelCor(resamples(model\_list)) #show model correlation

## glm rpart ranger nnet  
## glm 1.0000000 0.2040092 0.6490214 0.4734511  
## rpart 0.2040092 1.0000000 0.5115144 0.2942249  
## ranger 0.6490214 0.5115144 1.0000000 0.4723764  
## nnet 0.4734511 0.2942249 0.4723764 1.0000000

### Creating the Ensemble list

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

### Examine the ensemble

summary(ensemble)

## The following models were ensembled: glm, rpart, ranger, nnet   
## They were weighted:   
## 2.3947 -1.3034 0.261 -3.9062 0.1521  
## The resulting ROC is: 0.7138  
## The fit for each individual model on the ROC is:   
## method ROC ROCSD  
## glm 0.7090144 0.01406007  
## rpart 0.6397444 0.02400130  
## ranger 0.7320345 0.01606727  
## nnet 0.6836972 0.02525353

#training set  
pred\_ensemble = predict(ensemble, train, type = "raw")  
confusionMatrix(pred\_ensemble,train$Class)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 320 0  
## Yes 1 633  
##   
## Accuracy : 0.999   
## 95% CI : (0.9942, 1)  
## No Information Rate : 0.6635   
## P-Value [Acc > NIR] : <2e-16   
##   
## Kappa : 0.9977   
##   
## Mcnemar's Test P-Value : 1   
##   
## Sensitivity : 0.9969   
## Specificity : 1.0000   
## Pos Pred Value : 1.0000   
## Neg Pred Value : 0.9984   
## Prevalence : 0.3365   
## Detection Rate : 0.3354   
## Detection Prevalence : 0.3354   
## Balanced Accuracy : 0.9984   
##   
## 'Positive' Class : No   
##

#testing set  
pred\_ensemble\_test = predict(ensemble, test, type = "raw")  
confusionMatrix(pred\_ensemble\_test,test$Class)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 37 44  
## Yes 100 227  
##   
## Accuracy : 0.6471   
## 95% CI : (0.5985, 0.6934)  
## No Information Rate : 0.6642   
## P-Value [Acc > NIR] : 0.7848   
##   
## Kappa : 0.1198   
##   
## Mcnemar's Test P-Value : 4.576e-06   
##   
## Sensitivity : 0.27007   
## Specificity : 0.83764   
## Pos Pred Value : 0.45679   
## Neg Pred Value : 0.69419   
## Prevalence : 0.33578   
## Detection Rate : 0.09069   
## Detection Prevalence : 0.19853   
## Balanced Accuracy : 0.55386   
##   
## 'Positive' Class : No   
##

### Ensemble of list

modelCor(resamples(model\_list)) #show model correlation

## ranger nn glm rpart  
## ranger 1.0000000 0.6253524 0.6287914 0.6512695  
## nn 0.6253524 1.0000000 0.9281034 0.2134394  
## glm 0.6287914 0.9281034 1.0000000 0.2040092  
## rpart 0.6512695 0.2134394 0.2040092 1.0000000

### Stacking

control2 = trainControl(  
 method = "cv",  
 number = 3, #to save time, we'll use 3 fold cross-validation rather than 10  
 savePredictions = "final",  
 classProbs = TRUE, #instructs caret to calculate probabilities (rather than providing final classifications)  
 summaryFunction = twoClassSummary, #enables calculation of AUC  
 index=createResample(train$Class), #new line needed (manages sampling in folds). Changed response variable to the correct dataset  
 verboseIter = TRUE  
 )

**Below took 16 min to finish**

modelCor(resamples(model\_list3))

## ranger1 glm rpart  
## ranger1 1.0000000 0.6172611 0.2823697  
## glm 0.6172611 1.0000000 0.4260718  
## rpart 0.2823697 0.4260718 1.0000000

### Building the ensemble

ensemble3 = caretEnsemble(  
 model\_list3,   
 metric="ROC",  
 trControl=control2)

## + Resample01: parameter=none   
## - Resample01: parameter=none   
## + Resample02: parameter=none   
## - Resample02: parameter=none   
## + Resample03: parameter=none   
## - Resample03: parameter=none   
## + Resample04: parameter=none   
## - Resample04: parameter=none   
## + Resample05: parameter=none   
## - Resample05: parameter=none   
## + Resample06: parameter=none   
## - Resample06: parameter=none   
## + Resample07: parameter=none   
## - Resample07: parameter=none   
## + Resample08: parameter=none   
## - Resample08: parameter=none   
## + Resample09: parameter=none   
## - Resample09: parameter=none   
## + Resample10: parameter=none   
## - Resample10: parameter=none   
## Aggregating results  
## Fitting final model on full training set

### Examine the ensemble\_3

summary(ensemble3)

## The following models were ensembled: ranger1, glm, rpart   
## They were weighted:   
## 2.5645 -4.3011 -0.8438 0.1479  
## The resulting ROC is: 0.7174  
## The fit for each individual model on the ROC is:   
## method ROC ROCSD  
## ranger1 0.7387508 0.02743596  
## glm 0.7047481 0.02614008  
## rpart 0.6434663 0.04554155

#training set  
pred\_ensemble3 = predict(ensemble3, train, type = "raw")  
confusionMatrix(pred\_ensemble3,train$Class)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 318 0  
## Yes 3 633  
##   
## Accuracy : 0.9969   
## 95% CI : (0.9908, 0.9994)  
## No Information Rate : 0.6635   
## P-Value [Acc > NIR] : <2e-16   
##   
## Kappa : 0.9929   
##   
## Mcnemar's Test P-Value : 0.2482   
##   
## Sensitivity : 0.9907   
## Specificity : 1.0000   
## Pos Pred Value : 1.0000   
## Neg Pred Value : 0.9953   
## Prevalence : 0.3365   
## Detection Rate : 0.3333   
## Detection Prevalence : 0.3333   
## Balanced Accuracy : 0.9953   
##   
## 'Positive' Class : No   
##

#testing set  
pred\_ensemble\_test3 = predict(ensemble3, test, type = "raw")  
confusionMatrix(pred\_ensemble\_test3,test$Class)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 37 37  
## Yes 100 234  
##   
## Accuracy : 0.6642   
## 95% CI : (0.6161, 0.7099)  
## No Information Rate : 0.6642   
## P-Value [Acc > NIR] : 0.5232   
##   
## Kappa : 0.1507   
##   
## Mcnemar's Test P-Value : 1.177e-07   
##   
## Sensitivity : 0.27007   
## Specificity : 0.86347   
## Pos Pred Value : 0.50000   
## Neg Pred Value : 0.70060   
## Prevalence : 0.33578   
## Detection Rate : 0.09069   
## Detection Prevalence : 0.18137   
## Balanced Accuracy : 0.56677   
##   
## 'Positive' Class : No   
##

### stacking

start\_time = Sys.time() #Put here to measure how long this code takes to run  
  
stack2 = caretStack(  
 model\_list3, #use the list of models already specified  
 method ="glm", #stack models linearly  
 metric ="ROC", #maximize AUC  
 ###DO NOT use same trControl object here as you used to construct models  
 trControl=trainControl(  
 method="cv",  
 number=10,  
 savePredictions="final",  
 classProbs=TRUE,  
 summaryFunction=twoClassSummary  
 )  
)  
end\_time = Sys.time()  
end\_time - start\_time

## Time difference of 1.426521 secs

### View of Stack

print(stack2)

## A glm ensemble of 3 base models: ranger1, glm, rpart  
##   
## Ensemble results:  
## Generalized Linear Model   
##   
## 3486 samples  
## 3 predictor  
## 2 classes: 'No', 'Yes'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 3138, 3137, 3137, 3137, 3138, 3138, ...   
## Resampling results:  
##   
## ROC Sens Spec   
## 0.7380952 0.3645461 0.8854426

#summary(stack2)

#### Stacked model to make predictions on the training and testing set.

#training set  
pred\_stack2 = predict(stack2, train, type = "raw")  
confusionMatrix(pred\_stack2,train$Class)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 318 0  
## Yes 3 633  
##   
## Accuracy : 0.9969   
## 95% CI : (0.9908, 0.9994)  
## No Information Rate : 0.6635   
## P-Value [Acc > NIR] : <2e-16   
##   
## Kappa : 0.9929   
##   
## Mcnemar's Test P-Value : 0.2482   
##   
## Sensitivity : 0.9907   
## Specificity : 1.0000   
## Pos Pred Value : 1.0000   
## Neg Pred Value : 0.9953   
## Prevalence : 0.3365   
## Detection Rate : 0.3333   
## Detection Prevalence : 0.3333   
## Balanced Accuracy : 0.9953   
##   
## 'Positive' Class : No   
##

#testing set  
pred\_stack\_test2 = predict(stack2, test, type = "raw")  
confusionMatrix(pred\_stack\_test2,test$Class)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 37 37  
## Yes 100 234  
##   
## Accuracy : 0.6642   
## 95% CI : (0.6161, 0.7099)  
## No Information Rate : 0.6642   
## P-Value [Acc > NIR] : 0.5232   
##   
## Kappa : 0.1507   
##   
## Mcnemar's Test P-Value : 1.177e-07   
##   
## Sensitivity : 0.27007   
## Specificity : 0.86347   
## Pos Pred Value : 0.50000   
## Neg Pred Value : 0.70060   
## Prevalence : 0.33578   
## Detection Rate : 0.09069   
## Detection Prevalence : 0.18137   
## Balanced Accuracy : 0.56677   
##   
## 'Positive' Class : No   
##

### XGBoost

train\_dummy = dummyVars(" ~ .", data = train) #creates dummy labels  
train\_xgb = data.frame(predict(train\_dummy, newdata = train)) #converts variables in dataset to dummies  
str(train\_xgb)

## 'data.frame': 954 obs. of 25 variables:  
## $ Class.No : num 0 0 0 0 0 0 0 0 0 1 ...  
## $ Class.Yes : num 1 1 1 1 1 1 1 1 1 0 ...  
## $ X.Revenue.Growth. : num 0.1115 0.1289 0.3735 0.0636 0.0421 ...  
## $ X.EPS.Diluted. : num 2.53 4.48 7.57 2.85 0.85 3.67 1.56 3.23 4.88 -0.02 ...  
## $ X.EBITDA.Margin. : num 0.31 0.456 0.531 0.355 0.438 0.248 0.323 0.312 0.172 0.039 ...  
## $ priceBookValueRatio : num 2.16 2.86 4.48 1.13 4.08 ...  
## $ debtEquityRatio : num 1.56 0.353 0 0.959 1.307 ...  
## $ debtRatio : num 0.444 0.206 0 0.332 0.44 ...  
## $ X.PE.ratio. : num 13.3 10.3 17.1 10 53.7 ...  
## $ Sector.Basic.Materials : num 0 0 0 0 0 0 0 0 0 0 ...  
## $ Sector.Communication.Services : num 0 0 0 1 0 0 0 0 0 0 ...  
## $ Sector.Consumer.Cyclical : num 1 0 0 0 0 0 0 0 0 0 ...  
## $ Sector.Consumer.Defensive : num 0 0 0 0 0 1 0 0 0 0 ...  
## $ Sector.Energy : num 0 0 0 0 0 0 0 0 1 0 ...  
## $ Sector.Financial.Services : num 0 0 0 0 0 0 0 0 0 0 ...  
## $ Sector.Healthcare : num 0 0 0 0 0 0 0 0 0 0 ...  
## $ Sector.Industrials : num 0 0 0 0 0 0 0 0 0 0 ...  
## $ Sector.Real.Estate : num 0 0 0 0 0 0 0 0 0 0 ...  
## $ Sector.Technology : num 0 1 1 0 1 0 1 1 0 1 ...  
## $ Sector.Utilities : num 0 0 0 0 0 0 0 0 0 0 ...  
## $ X.5Y.Revenue.Growth..per.Share..: num 0.1094 0.077 0.4281 -0.0081 0.0416 ...  
## $ returnOnAssets : num 0.303 0.344 0.325 0.143 0.057 ...  
## $ returnOnEquity : num 0.1638 0.2824 0.2628 0.1052 0.0774 ...  
## $ returnOnCapitalEmployed : num 0.0531 0.1444 0.3165 0.0352 0.1495 ...  
## $ quickRatio : num 0.54 1.105 6.94 0.492 3.786 ...

test\_dummy = dummyVars(" ~ .", data = test) #creates dummy labels  
test\_xgb = data.frame(predict(test\_dummy, newdata = test)) #converts variables in dataset to dummies

train\_xgb = train\_xgb %>% dplyr::select(-Class.No) #NOTE! select conflict  
test\_xgb = test\_xgb %>% dplyr::select(-Class.No)

# Tuning model

start\_time = Sys.time() #for timing  
  
set.seed(999)  
ctrl = trainControl(method = "cv",  
 number = 5) #10 fold, k-fold cross-validation  
  
tgrid = expand.grid(  
 nrounds = 100, #50, 100, and 150 in default tuning  
 max\_depth = c(1,2,3,4), #1, 2, and 3 in default tuning  
 eta = c(0.01, 0.1, 0.2, 0.3), #0.3 and 0.4 in default tuning  
 gamma = 0, #fixed at 0 in default tuning  
 colsample\_bytree = c(0.6, 0.8, 1), #0.6 and 0.6 in default tuning  
 min\_child\_weight = 1, #fixed at 1 in default tuning  
 subsample = c(0.8, 1) #0.5, 0.75, and 1 in default tuning, we don't have much data so can choose a larger value  
)  
  
fitxgb2 = train(as.factor(Class.Yes)~.,  
 data = train\_xgb,  
 method="xgbTree",  
 tuneGrid = tgrid,  
 trControl=ctrl)  
  
end\_time = Sys.time()  
end\_time-start\_time

## Time difference of 3.868898 mins

saveRDS(fitxgb2,"fitxgb2.rds")  
rm(fitxgb2)

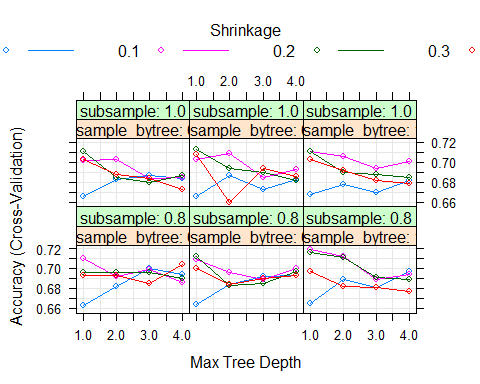
fitxgb2 = readRDS("fitxgb2.rds")

fitxgb2

## eXtreme Gradient Boosting   
##   
## 954 samples  
## 23 predictor  
## 2 classes: '0', '1'   
##   
## No pre-processing  
## Resampling: Cross-Validated (5 fold)   
## Summary of sample sizes: 763, 763, 763, 763, 764

## Tuning parameter 'nrounds' was held constant at a value of 100  
## Tuning  
## parameter 'gamma' was held constant at a value of 0  
## Tuning  
## parameter 'min\_child\_weight' 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 nrounds = 100, max\_depth = 1, eta  
## = 0.1, gamma = 0, colsample\_bytree = 1, min\_child\_weight = 1 and subsample  
## = 0.8.

plot(fitxgb2)



predxgbtrain2 = predict(fitxgb2, train\_xgb)  
confusionMatrix(as.factor(train\_xgb$Class.Yes), predxgbtrain2,positive="1")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 116 205  
## 1 45 588  
##   
## Accuracy : 0.7379   
## 95% CI : (0.7088, 0.7656)  
## No Information Rate : 0.8312   
## P-Value [Acc > NIR] : 1   
##   
## Kappa : 0.3309   
##   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 0.7415   
## Specificity : 0.7205   
## Pos Pred Value : 0.9289   
## Neg Pred Value : 0.3614   
## Prevalence : 0.8312   
## Detection Rate : 0.6164   
## Detection Prevalence : 0.6635   
## Balanced Accuracy : 0.7310   
##   
## 'Positive' Class : 1   
##

predxgbtest2 = predict(fitxgb2, test\_xgb)  
confusionMatrix(as.factor(test\_xgb$Class.Yes), predxgbtest2,positive="1")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 32 105  
## 1 33 238  
##   
## Accuracy : 0.6618   
## 95% CI : (0.6136, 0.7076)  
## No Information Rate : 0.8407   
## P-Value [Acc > NIR] : 1   
##   
## Kappa : 0.1285   
##   
## Mcnemar's Test P-Value : 1.504e-09   
##   
## Sensitivity : 0.6939   
## Specificity : 0.4923   
## Pos Pred Value : 0.8782   
## Neg Pred Value : 0.2336   
## Prevalence : 0.8407   
## Detection Rate : 0.5833   
## Detection Prevalence : 0.6642   
## Balanced Accuracy : 0.5931   
##   
## 'Positive' Class : 1   
##