|  |
| --- |
| ## Import dataset and needed R libraries |
|  |

|  |
| --- |
| library(ggplot2) |
|  |

|  |
| --- |
| library(dplyr) |
|  |

|  |
| --- |
| library(pROC) ## plot ROC curves |
|  |

|  |
| --- |
| library(MASS) ## Package for stepwise regression using stepAIC() |
|  |

|  |
| --- |
| library(reshape2) ## Correlation Matrix w/ Heatmat |
|  |

|  |
| --- |
| library(rpart) # decision tree methodology |
|  |

|  |
| --- |
| library(rpart.plot) # decision tree visualization |
|  |

|  |
| --- |
| library(randomForest) # random forest methodology |
|  |

|  |
| --- |
| library(gbm) # Boosting regression |
|  |

|  |
| --- |
| library(h2o) # Deep learning package, requires JAVA |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| bankdata <- read.csv("bank-additional/bank-additional-full.csv", header = TRUE, sep = ';') |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| ### -------------------------------------------------------------------------------- ### |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| ## Data Preprocessing/Cleaning  |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| names(bankdata) |
|  |

|  |
| --- |
| colSums(is.na(bankdata)) # checks for NA values |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| # removes "unknown" values |
|  |

|  |
| --- |
| bankdata <- subset(bankdata, job!="unknown") |
|  |

|  |
| --- |
| bankdata <- subset(bankdata, marital!="unknown") |
|  |

|  |
| --- |
| bankdata <- subset(bankdata, education!="unknown") |
|  |

|  |
| --- |
| bankdata <- subset(bankdata, default!="unknown") |
|  |

|  |
| --- |
| bankdata <- subset(bankdata, housing!="unknown") |
|  |

|  |
| --- |
| bankdata <- subset(bankdata, loan!="unknown") |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| ### -------------------------------------------------------------------------------- ### |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| ## EDA and Data Visualization |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| ggplot(bankdata, aes(y)) + geom\_bar(aes(y = (..count..)/sum(..count..),fill = y)) + |
|  |

|  |
| --- |
| scale\_y\_continuous(labels = scales::percent) + theme\_classic() + |
|  |

|  |
| --- |
| labs(title = "No vs Yes (Response)", y = "Percent") |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| ggplot(bankdata, aes(x = age)) + geom\_histogram(binwidth = 5, col = "black", fill = "pink") +  |
|  |

|  |
| --- |
| theme\_classic() |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| ggplot(bankdata, aes(x = y, y = age, fill = y)) + geom\_boxplot() |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| ggplot(bankdata, aes(x = y, y = emp.var.rate, fill = y)) + geom\_boxplot() + theme\_classic() |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| ggplot(bankdata, aes(x = y, y = nr.employed, fill = y)) + geom\_boxplot() + theme\_classic() |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| ggplot(bankdata, aes(x = y, y = euribor3m, fill = y)) + geom\_boxplot() + |
|  |

|  |
| --- |
| theme\_classic() |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| ggplot(data=bankdata, aes(x=job, fill=job)) + geom\_bar(stat="count") +  |
|  |

|  |
| --- |
| theme\_classic() + |
|  |

|  |
| --- |
| theme(axis.text.x = element\_text(angle = 90)) + facet\_grid(rows = vars(y)) + |
|  |

|  |
| --- |
| labs(title = "Job by Response", x = "Job", y = "Number of Samples") |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| ggplot(data=bankdata, aes(x=marital, fill=marital)) + geom\_bar(stat="count") +  |
|  |

|  |
| --- |
| theme\_classic() + |
|  |

|  |
| --- |
| theme(axis.text.x = element\_text(angle = 90)) + facet\_grid(cols = vars(y)) + |
|  |

|  |
| --- |
| labs(title = "Marital Status by Response", x = "Marital Status", y = "Number of Samples") |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| ggplot(data=bankdata, aes(x=education, fill=education)) + geom\_bar(stat="count") +  |
|  |

|  |
| --- |
| theme\_classic() + |
|  |

|  |
| --- |
| theme(axis.text.x = element\_text(angle = 90)) + facet\_grid(rows = vars(y)) + |
|  |

|  |
| --- |
| labs(title = "Educational Level by Response", x = "Educational Level", y = "Number of Samples") |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| ggplot(bankdata, aes(day\_of\_week)) + geom\_bar(aes(y = (..count..)/sum(..count..),fill = y)) + |
|  |

|  |
| --- |
| scale\_y\_continuous(labels = scales::percent) + theme\_classic() + |
|  |

|  |
| --- |
| labs(title = "Day of Week by Response", y = "Percent") + facet\_grid(cols = vars(y)) |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| ggplot(bankdata, aes(poutcome)) + geom\_bar(aes(y = (..count..)/sum(..count..),fill = y)) + |
|  |

|  |
| --- |
| scale\_y\_continuous(labels = scales::percent) + theme\_classic() + |
|  |

|  |
| --- |
| labs(title = "Previous Outcome of Campaign with Response", y = "Percent") +  |
|  |

|  |
| --- |
| facet\_grid(cols = vars(y)) |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| ## Convert all categorical predictor variables to numeric/dummy variables for Logistic |
|  |

|  |
| --- |
| ## Regression and Correlation Plot |
|  |

|  |
| --- |
| bankdata$job <- as.numeric(as.factor(bankdata$job)) |
|  |

|  |
| --- |
| bankdata$marital <- as.numeric(as.factor(bankdata$marital)) |
|  |

|  |
| --- |
| bankdata$education <- as.numeric(as.factor(bankdata$education)) |
|  |

|  |
| --- |
| bankdata$month <- as.numeric(as.factor(bankdata$month)) |
|  |

|  |
| --- |
| bankdata$contact <- as.numeric(as.factor(bankdata$contact)) |
|  |

|  |
| --- |
| bankdata$poutcome <- as.numeric(as.factor(bankdata$poutcome)) |
|  |

|  |
| --- |
| bankdata$day\_of\_week <- as.numeric(as.factor(bankdata$day\_of\_week)) |
|  |

|  |
| --- |
| bankdata$default<- ifelse(bankdata$default == "yes", 1, 0) |
|  |

|  |
| --- |
| bankdata$housing <- ifelse(bankdata$housing== "yes", 1, 0) |
|  |

|  |
| --- |
| bankdata$loan<- ifelse(bankdata$loan== "yes", 1, 0) |
|  |

|  |
| --- |
| bankdata$y <- ifelse(bankdata$y== "yes", 1, 0) |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| #correlation matrix w/ heatmap (requires all var to be numeric) |
|  |

|  |
| --- |
| corr <- cor(bankdata) |
|  |

|  |
| --- |
| melt\_corr <- melt(corr) |
|  |

|  |
| --- |
| ggplot(data = melt\_corr, aes(x=Var1, y=Var2, fill=value)) +  |
|  |

|  |
| --- |
| geom\_tile(color = "white")+ |
|  |

|  |
| --- |
| scale\_fill\_gradient2(low = "blue", high = "red", mid = "white",  |
|  |

|  |
| --- |
| midpoint = 0, limit = c(-1,1), space = "Lab",name="Pearson\nCorrelation") +  |
|  |

|  |
| --- |
| theme(axis.text.x = element\_text(angle = 90)) |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| ### -------------------------------------------------------------------------------- ### |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| ## Logistic Regression |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| ### -------------------------------------------------------------------------------- ### |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| # Split Sample Validation: 80% train / 20% test |
|  |

|  |
| --- |
| set.seed(1) |
|  |

|  |
| --- |
| train\_bank <- sample(1:nrow(bankdata),0.8\*nrow(bankdata)) # using 80% as training data |
|  |

|  |
| --- |
| train <- bankdata[train\_bank,] |
|  |

|  |
| --- |
| test <- bankdata[-train\_bank,] |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| # LR Full Model using all features in the data  |
|  |

|  |
| --- |
| bank.log\_full <- glm(y ~ ., data = train, family = "binomial") |
|  |

|  |
| --- |
| summary(bank.log\_full) # AIC: 11587 |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| # Estimate probabilities for each sample |
|  |

|  |
| --- |
| full.mod.probs <- predict(bank.log\_full, test[,1:20], type = "response") |
|  |

|  |
| --- |
| # Predict classifications. If probability is above threshold, classify as y = 1 |
|  |

|  |
| --- |
| full.mod.predclasses <- ifelse(full.mod.probs > 0.5, 1, 0) |
|  |

|  |
| --- |
| actual.classes <- test$y |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| # Confusion Matrix |
|  |

|  |
| --- |
| full.confmatrix <- table(predicted = full.mod.predclasses, actual = actual.classes) |
|  |

|  |
| --- |
| full.confmatrix |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| # Classification Accuracy  |
|  |

|  |
| --- |
| full.accuracy <- (full.confmatrix[1,1] + full.confmatrix[2,2])/sum(full.confmatrix) |
|  |

|  |
| --- |
| full.accuracy\*100 |
|  |

|  |
| --- |
| ## 90.4%  |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| # True Positive Rate |
|  |

|  |
| --- |
| # Calculated as True Positives / True Positives + True negatives |
|  |

|  |
| --- |
| full.TPR.log <- full.confmatrix[2,2]/(full.confmatrix[2,2]+full.confmatrix[1,2]) |
|  |

|  |
| --- |
| full.TPR.log\*100 |
|  |

|  |
| --- |
| # 39.92%  |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| # False Positive Rate |
|  |

|  |
| --- |
| full.FPR.log <- full.confmatrix[2,1]/(full.confmatrix[2,1]+full.confmatrix[1,1]) |
|  |

|  |
| --- |
| full.FPR.log\*100 |
|  |

|  |
| --- |
| # 2.49%  |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| # ROC LR Full Model |
|  |

|  |
| --- |
| par(pty = "s") # condenses the axes on the plot to be visually easier to read |
|  |

|  |
| --- |
| # calculate and plot ROC and AUC |
|  |

|  |
| --- |
| roc(test$y, full.mod.probs, plot = TRUE, legacy.axes = TRUE, percent = TRUE, |
|  |

|  |
| --- |
| xlab = "100 - Specificity % (FPR)", ylab = "Sensitivity % (TPR)", |
|  |

|  |
| --- |
| col = "darkolivegreen", lwd = 3, print.auc = TRUE, print.auc.x = 45) |
|  |

|  |
| --- |
| # label the ROC plot |
|  |

|  |
| --- |
| legend("bottomright", legend="Full Logistic", col = "darkolivegreen", lwd = 3) |
|  |

|  |
| --- |
| ## AUC of 92.7% |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| ## Using the MASS library package, use stepAIC function |
|  |

|  |
| --- |
| # Stepwise variable selection/Backwards stepwise Regression |
|  |

|  |
| --- |
| set.seed(1) |
|  |

|  |
| --- |
| # stepAIC() takes our trainedm, full model does backwards regression by default |
|  |

|  |
| --- |
| # so at each step, one predictor variable is removed |
|  |

|  |
| --- |
| stepwise.mod <- stepAIC(bank.log\_full, trace = FALSE) |
|  |

|  |
| --- |
| summary(stepwise.mod) # AIC: 11579 |
|  |

|  |
| --- |
| coef(stepwise.mod) |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| ## Using the features stepAIC() chose for us, we train a new reduced LR model on those chosen |
|  |

|  |
| --- |
| # features |
|  |

|  |
| --- |
| reduced.mod <- glm(formula = y ~ job + marital + education + contact + month +  |
|  |

|  |
| --- |
| day\_of\_week + duration + campaign + pdays + poutcome + emp.var.rate +  |
|  |

|  |
| --- |
| cons.price.idx + euribor3m + nr.employed, family = "binomial",  |
|  |

|  |
| --- |
| data = train) |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| summary(reduced.mod) |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| ## Estimated probabilities for each sample in test data set  |
|  |

|  |
| --- |
| reduced.probs <- predict(reduced.mod, test[,1:20], type = "response") |
|  |

|  |
| --- |
| ## Predict classifications from each probability |
|  |

|  |
| --- |
| reduced.class <- ifelse(reduced.probs > 0.5, 1, 0) |
|  |

|  |
| --- |
| actual.class <- test$y |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| # Confusion Matrix |
|  |

|  |
| --- |
| reduced.confmatrix <- table(predicted = reduced.class, actual = actual.class) |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| # Classification Accuracy  |
|  |

|  |
| --- |
| reduced.mod.accuracy <- (reduced.confmatrix[1,1] + reduced.confmatrix[2,2])/sum(reduced.confmatrix) |
|  |

|  |
| --- |
| reduced.mod.accuracy |
|  |

|  |
| --- |
| ## Accuracy: 90.4%, same as full model |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| # ROC Reduced Model |
|  |

|  |
| --- |
| par(pty = "s") # condenses the axes on the plot to be visually easier to read |
|  |

|  |
| --- |
| # calculate and plot ROC and AUC |
|  |

|  |
| --- |
| roc(test$y, reduced.probs, plot = TRUE, legacy.axes = TRUE, percent = TRUE,  |
|  |

|  |
| --- |
| xlab = "100 - Specificity % (FPR)", ylab = "Sensitivity % (TPR)", |
|  |

|  |
| --- |
| col = "deepskyblue3", lwd = 3, print.auc = TRUE, print.auc.x = 45) |
|  |

|  |
| --- |
| # label the ROC plot |
|  |

|  |
| --- |
| legend("bottomright", legend="Reduced Logistic", col = "deepskyblue3", lwd = 3) |
|  |

|  |
| --- |
| ## AUC of 92.7%, same as the full model  |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| ### -------------------------------------------------------------------------------- ### |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| ## Random Forest |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| ### -------------------------------------------------------------------------------- ### |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| ## Reread dataset into directory, remove unknown values again, and convert all categorical |
|  |

|  |
| --- |
| # variables to factors, inorder to be used as input into RF  |
|  |

|  |
| --- |
| ## RF won't take the dummy variables from LR as input  |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| bankdata <- read.csv("~/Desktop/Fall 2020 Quarter/STA141A/Final Project/BankData/bank-additional/bank-additional-full.csv", sep=";", stringsAsFactors = TRUE) |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| # removes "unknown" values |
|  |

|  |
| --- |
| bankdata <- subset(bankdata, job!="unknown") |
|  |

|  |
| --- |
| bankdata <- subset(bankdata, marital!="unknown") |
|  |

|  |
| --- |
| bankdata <- subset(bankdata, education!="unknown") |
|  |

|  |
| --- |
| bankdata <- subset(bankdata, default!="unknown") |
|  |

|  |
| --- |
| bankdata <- subset(bankdata, housing!="unknown") |
|  |

|  |
| --- |
| bankdata <- subset(bankdata, loan!="unknown") |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| #convert all categorical variables to factors, inorder to be used as input into RF  |
|  |

|  |
| --- |
| bankdata$job <- as.factor(bankdata$job) |
|  |

|  |
| --- |
| bankdata$marital <- as.factor(bankdata$marital) |
|  |

|  |
| --- |
| bankdata$education <- as.factor(bankdata$education) |
|  |

|  |
| --- |
| bankdata$month <- as.factor(bankdata$month) |
|  |

|  |
| --- |
| bankdata$contact <- as.factor(bankdata$contact) |
|  |

|  |
| --- |
| bankdata$poutcome <- as.factor(bankdata$poutcome) |
|  |

|  |
| --- |
| bankdata$day\_of\_week <- as.factor(bankdata$day\_of\_week) |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| # training/test, set seed to 1 to ensure we are using the same train and test throughout the whole |
|  |

|  |
| --- |
| # code across all models  |
|  |

|  |
| --- |
| set.seed(1) |
|  |

|  |
| --- |
| train\_bank <- sample(1:nrow(bankdata),0.8\*nrow(bankdata)) # using 80% as training data |
|  |

|  |
| --- |
| train <- bankdata[train\_bank,] |
|  |

|  |
| --- |
| test <- bankdata[-train\_bank,] |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| ## Train RF model on m = 4 |
|  |

|  |
| --- |
| set.seed(1) |
|  |

|  |
| --- |
| rf.bankdata.4 = randomForest(y ~ ., data = train, mtry = 4, importance = T) |
|  |

|  |
| --- |
| rf.bankdata.4 |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| # plot to see the best choice of number of trees  |
|  |

|  |
| --- |
| plot(rf.bankdata.4) |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| ## Train RF model on m = 5 |
|  |

|  |
| --- |
| set.seed(1) |
|  |

|  |
| --- |
| rf.bankdata.5 = randomForest(y ~ ., data = train, mtry = 5, importance = T) |
|  |

|  |
| --- |
| rf.bankdata.5 |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| # plot to see the best choice of number of trees  |
|  |

|  |
| --- |
| plot(rf.bankdata.5) |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| ## Predict classes using our trained RF model m = 4  |
|  |

|  |
| --- |
| rf.test.4 <- predict(rf.bankdata.4, test[,1:20], type = "response") |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| # Confusion matrix for RF with m = 4 |
|  |

|  |
| --- |
| rf4\_conf.matrix <- table(predicted = rf.test.4, actual = test$y) |
|  |

|  |
| --- |
| names(rf4\_conf.matrix) = paste("Confusion matrix when m = 4:",sep = " ") |
|  |

|  |
| --- |
| rf4\_conf.matrix |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| # Classification Accuracy when m = 4 |
|  |

|  |
| --- |
| set.seed(1) |
|  |

|  |
| --- |
| rf.accuracy.4 <- (rf4\_conf.matrix[1,1] + rf4\_conf.matrix[2,2])/sum(rf4\_conf.matrix) |
|  |

|  |
| --- |
| names(rf.accuracy.4) = paste("Accuracy when m = 4:",sep = " ") |
|  |

|  |
| --- |
| rf.accuracy.4\*100 |
|  |

|  |
| --- |
| ## 91.4 %  |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| ## Predict classes using our trained RF model m = 5 |
|  |

|  |
| --- |
| rf.test.5 <- predict(rf.bankdata.5, test[,1:20], type = "response") |
|  |

|  |
| --- |
| # Confusion matrix for RF with m = 5 |
|  |

|  |
| --- |
| rf5\_conf.matrix <- table(predicted = rf.test.5, actual = test$y) |
|  |

|  |
| --- |
| names(rf5\_conf.matrix) = paste("Confusion matrix when m = 5:",sep = " ") |
|  |

|  |
| --- |
| rf5\_conf.matrix |
|  |

|  |
| --- |
| # 91.3% |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| ## For loop to calculate OOB error on train set and error on test set to choose the best m  |
|  |

|  |
| --- |
| oob.err = double(15) #Out-of-bag error |
|  |

|  |
| --- |
| test.err = double(15) #Test error |
|  |

|  |
| --- |
| for(mtry in 1:15){ |
|  |

|  |
| --- |
| fit.train = randomForest(as.factor(y) ~ ., data = train, mtry = mtry, ntree = 50)  |
|  |

|  |
| --- |
| oob.err[mtry] = fit.train$err.rate[1] |
|  |

|  |
| --- |
| fit.test = randomForest(as.factor(y) ~ ., data = test, mtry = mtry, ntree = 50) |
|  |

|  |
| --- |
| test.err[mtry] = fit.test$err.rate[1] |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
| matplot(1:mtry, cbind(test.err, oob.err), pch = 23, col = c("red", "blue"),  |
|  |

|  |
| --- |
| type = "b", ylab="Error",xlab="m") |
|  |

|  |
| --- |
| legend("topright", legend = c("OOB", "Test"), pch = 23, col = c("blue", "red")) |
|  |

|  |
| --- |
| ## choose the m split that gives the lowest OOB error (if we want our model to be based off accuracy) |
|  |

|  |
| --- |
| ## or choose the m split that will give the highest AUC value (if we want our model to be based off sensisitivity) |
|  |

|  |
| --- |
| #cbind(test.err, oob.err) |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| ## The plot shows relative low error for OOB and test error when m = 2 |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| ## Now we Train RF model on m = 2 |
|  |

|  |
| --- |
| set.seed(1) |
|  |

|  |
| --- |
| rf.bankdata.2 = randomForest(y ~ ., data= train, mtry = 2, ntree = 50) |
|  |

|  |
| --- |
| rf.bankdata.2 |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| ## Predict classifications for m = 2 |
|  |

|  |
| --- |
| rf.test.2 <- predict(rf.bankdata.2, test[,1:20], type = "response") |
|  |

|  |
| --- |
| # Confusion matrix for RF with m = 2 |
|  |

|  |
| --- |
| rf2\_conf.matrix <- table(predicted = rf.test.2, actual = test$y) |
|  |

|  |
| --- |
| names(rf2\_conf.matrix) = paste("Confusion matrix when m = 2:",sep = " ") |
|  |

|  |
| --- |
| rf2\_conf.matrix |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| ## Classification accuracy |
|  |

|  |
| --- |
| rf.accuracy.2 <- (rf2\_conf.matrix[1,1] + rf2\_conf.matrix[2,2])/sum(rf2\_conf.matrix) |
|  |

|  |
| --- |
| names(rf.accuracy.2) = paste("Accuracy when m = 2:",sep = " ") |
|  |

|  |
| --- |
| rf.accuracy.2\*100 |
|  |

|  |
| --- |
| # 90.7%  |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| ## we find m = 4 is the best choice, highest accuracy and still low computationally intensive  |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| ## Plot ROC and AUC for RF when m = 4  |
|  |

|  |
| --- |
| ## Change the type of output to "vote" because the roc function needs that to calculate ROC values |
|  |

|  |
| --- |
| rf.test.4 <- predict(rf.bankdata.4, test[,1:20], type = "vote") |
|  |

|  |
| --- |
| ## Convert the array to a dataframe otherwise ROC will give error |
|  |

|  |
| --- |
| rf\_df.test4 <- as.data.frame(rf.test.4) |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| ## ROC Curve for RF m = 4 |
|  |

|  |
| --- |
| par(pty = "s") |
|  |

|  |
| --- |
| roc(test$y, rf\_df.test4[,1], plot = TRUE, legacy.axes = TRUE, percent = TRUE,  |
|  |

|  |
| --- |
| xlab = "100 - Specificity % (FPR)", ylab = "Sensitivity % (TPR)", |
|  |

|  |
| --- |
| col = "darkmagenta", lwd = 3, print.auc = TRUE, print.auc.x = 45)  |
|  |

|  |
| --- |
| legend("bottomright", legend="RF: m = 4", col = "darkmagenta", lwd = 3) |
|  |

|  |
| --- |
| ## AUC for RF with m = 4 is 94.8% |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| # variable importance plot on our trained random forest model, with m = 4 |
|  |

|  |
| --- |
| # duration var seems like it is too strong of a predictor, may be giving us falsely high AUC  |
|  |

|  |
| --- |
| # and accuracy  |
|  |

|  |
| --- |
| varImpPlot(rf.bankdata.4) |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| ### -------------------------------------------------------------------------------- ### |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| # Now we build new LR and RF models removing duration variable to see how our accuracy and  |
|  |

|  |
| --- |
| # AUC change  |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| ### -------------------------------------------------------------------------------- ### |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| ## New Logisitic Regression model removing duration var |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| # Reread in data into directory |
|  |

|  |
| --- |
| bankdata <- read.csv("~/Desktop/Fall 2020 Quarter/STA141A/Final Project/BankData/bank-additional/bank-additional-full.csv", sep=";", stringsAsFactors = TRUE) |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| # remove "unknown" values from our dataset, this time REMOVE DURATION |
|  |

|  |
| --- |
| bankdata <- subset(bankdata, job!="unknown") |
|  |

|  |
| --- |
| bankdata <- subset(bankdata, marital!="unknown") |
|  |

|  |
| --- |
| bankdata <- subset(bankdata, education!="unknown") |
|  |

|  |
| --- |
| bankdata <- subset(bankdata, default!="unknown") |
|  |

|  |
| --- |
| bankdata <- subset(bankdata, housing!="unknown") |
|  |

|  |
| --- |
| bankdata <- subset(bankdata, loan!="unknown") |
|  |

|  |
| --- |
| bankdata <- subset(bankdata, select = -c(duration)) # removing duration |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| ## Convert all categorical predictor variables to numeric, FOR LOGISTIC REGRESSION ONLY  |
|  |

|  |
| --- |
| bankdata$job <- as.numeric(as.factor(bankdata$job)) |
|  |

|  |
| --- |
| bankdata$marital <- as.numeric(as.factor(bankdata$marital)) |
|  |

|  |
| --- |
| bankdata$education <- as.numeric(as.factor(bankdata$education)) |
|  |

|  |
| --- |
| bankdata$month <- as.numeric(as.factor(bankdata$month)) |
|  |

|  |
| --- |
| bankdata$contact <- as.numeric(as.factor(bankdata$contact)) |
|  |

|  |
| --- |
| bankdata$poutcome <- as.numeric(as.factor(bankdata$poutcome)) |
|  |

|  |
| --- |
| bankdata$day\_of\_week <- as.numeric(as.factor(bankdata$day\_of\_week)) |
|  |

|  |
| --- |
| bankdata$default<- ifelse(bankdata$default == "yes", 1, 0) |
|  |

|  |
| --- |
| bankdata$housing <- ifelse(bankdata$housing== "yes", 1, 0) |
|  |

|  |
| --- |
| bankdata$loan<- ifelse(bankdata$loan== "yes", 1, 0) |
|  |

|  |
| --- |
| bankdata$y <- ifelse(bankdata$y== "yes", 1, 0) |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| # Split Sample Validation: 80% train / 20% test, set seed to 1 to ensure the same train and test |
|  |

|  |
| --- |
| # data through all the models |
|  |

|  |
| --- |
| set.seed(1) |
|  |

|  |
| --- |
| train\_bank <- sample(1:nrow(bankdata),0.8\*nrow(bankdata)) # using 80% as training data |
|  |

|  |
| --- |
| train <- bankdata[train\_bank,] |
|  |

|  |
| --- |
| test <- bankdata[-train\_bank,] |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| ## Train new logistic model with all predictors except duration predictor removed |
|  |

|  |
| --- |
| set.seed(1) |
|  |

|  |
| --- |
| new.glm <- glm(y ~ ., data = train, family = "binomial") |
|  |

|  |
| --- |
| ## Estimate probabilities of each observation in the test set |
|  |

|  |
| --- |
| new.probs <- predict(new.glm, test[,1:20], type = "response") |
|  |

|  |
| --- |
| ## Classify new samples from test data  |
|  |

|  |
| --- |
| new.classes <- ifelse(new.probs > 0.5, 1, 0) |
|  |

|  |
| --- |
| # see how our trained model performs on the test data without duration var |
|  |

|  |
| --- |
| new.conf.matrix <- table(predicted = new.classes, actual = test$y) |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| # Classification accuracy without duration var |
|  |

|  |
| --- |
| new.accuracy <- (new.conf.matrix[1,1] + new.conf.matrix[2,2])/sum(new.conf.matrix) |
|  |

|  |
| --- |
| #new.accuracy |
|  |

|  |
| --- |
| # 89.0% |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| # True Positive Rate |
|  |

|  |
| --- |
| new.TPR <- new.conf.matrix[2,2]/(new.conf.matrix[2,2]+new.conf.matrix[1,2]) |
|  |

|  |
| --- |
| #new.TPR |
|  |

|  |
| --- |
| # 21.9% |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| # False Positive Rate |
|  |

|  |
| --- |
| new.FPR <- new.conf.matrix[2,1]/(new.conf.matrix[2,1]+new.conf.matrix[1,1]) |
|  |

|  |
| --- |
| #new.FPR |
|  |

|  |
| --- |
| # 1.52% |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| ## Perform stepwise regression on new trained model to select the best linear combination of predictors |
|  |

|  |
| --- |
| set.seed(1) |
|  |

|  |
| --- |
| new.full.glm <- glm(y ~ ., data = train, family = "binomial") |
|  |

|  |
| --- |
| new.stepwise <- stepAIC(new.full.glm, trace = FALSE) |
|  |

|  |
| --- |
| summary(new.stepwise) |
|  |

|  |
| --- |
| ## New reduced model without duration var |
|  |

|  |
| --- |
| new.reduced.glm <- glm(formula = y ~ age + marital + education + contact + month +  |
|  |

|  |
| --- |
| day\_of\_week + campaign + pdays + poutcome + emp.var.rate +  |
|  |

|  |
| --- |
| cons.price.idx + euribor3m + nr.employed, family = "binomial",  |
|  |

|  |
| --- |
| data = train) |
|  |

|  |
| --- |
| ## Estimate probabilities of each observation in the test set |
|  |

|  |
| --- |
| new.reduced.probs <- predict(new.reduced.glm, test[,1:20], type = "response") |
|  |

|  |
| --- |
| ## Classify new samples from test data  |
|  |

|  |
| --- |
| new.reduced.classes <- ifelse(new.reduced.probs > 0.5, 1, 0) |
|  |

|  |
| --- |
| # see how our trained model performs on the test data without duration var |
|  |

|  |
| --- |
| new.reduced.CF <- table(predicted = new.reduced.classes, actual = test$y) |
|  |

|  |
| --- |
| new.red.accuracy <- (new.reduced.CF[1,1] + new.reduced.CF[2,2])/sum(new.reduced.CF) |
|  |

|  |
| --- |
| new.red.accuracy |
|  |

|  |
| --- |
| # 89.0% |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| # True Positive Rate on reduced set of predictors |
|  |

|  |
| --- |
| new.red.TPR <- new.reduced.CF[2,2]/(new.reduced.CF[2,2]+new.reduced.CF[1,2]) |
|  |

|  |
| --- |
| #new.red.TPR |
|  |

|  |
| --- |
| # 21.9% |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| # False Positive Rate on reduced set of predictors  |
|  |

|  |
| --- |
| new.red.FPR <- new.reduced.CF[2,1]/(new.reduced.CF[2,1]+new.reduced.CF[1,1]) |
|  |

|  |
| --- |
| #new.red.FPR |
|  |

|  |
| --- |
| # 1.48% |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| ## our TPR is the same, which is good, and our FPR is a bit lower which is also good, thats what  |
|  |

|  |
| --- |
| # we want: high TPR and low FPR |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| # ROC Reduced LR Model removed duration var |
|  |

|  |
| --- |
| par(pty = "s") |
|  |

|  |
| --- |
| roc(test$y, new.reduced.probs, plot = TRUE, legacy.axes = TRUE, percent = TRUE,  |
|  |

|  |
| --- |
| xlab = "100 - Specificity % (FPR)", ylab = "Sensitivity % (TPR)", |
|  |

|  |
| --- |
| col = "deepskyblue3", lwd = 3, print.auc = TRUE, print.auc.x = 45) |
|  |

|  |
| --- |
| legend("bottomright", legend="New Reduced LR", col = "deepskyblue3", lwd = 3) |
|  |

|  |
| --- |
| roc(test$y, new.reduced.probs)  |
|  |

|  |
| --- |
| ## AUC 79.4 % |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| ### -------------------------------------------------------------------------------- ### |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| ## New Random Forest model removing duration var |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| bankdata <- read.csv("~/Desktop/Fall 2020 Quarter/STA141A/Final Project/BankData/bank-additional/bank-additional-full.csv", sep=";", stringsAsFactors = TRUE) |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| # removes "unknown" values |
|  |

|  |
| --- |
| bankdata <- subset(bankdata, job!="unknown") |
|  |

|  |
| --- |
| bankdata <- subset(bankdata, marital!="unknown") |
|  |

|  |
| --- |
| bankdata <- subset(bankdata, education!="unknown") |
|  |

|  |
| --- |
| bankdata <- subset(bankdata, default!="unknown") |
|  |

|  |
| --- |
| bankdata <- subset(bankdata, housing!="unknown") |
|  |

|  |
| --- |
| bankdata <- subset(bankdata, loan!="unknown") |
|  |

|  |
| --- |
| bankdata <- subset(bankdata, select = -c(duration)) # removing duration |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| # all categorical vars as factors for RF input |
|  |

|  |
| --- |
| bankdata$job <- as.factor(bankdata$job) |
|  |

|  |
| --- |
| bankdata$marital <- as.factor(bankdata$marital) |
|  |

|  |
| --- |
| bankdata$education <- as.factor(bankdata$education) |
|  |

|  |
| --- |
| bankdata$month <- as.factor(bankdata$month) |
|  |

|  |
| --- |
| bankdata$contact <- as.factor(bankdata$contact) |
|  |

|  |
| --- |
| bankdata$poutcome <- as.factor(bankdata$poutcome) |
|  |

|  |
| --- |
| bankdata$day\_of\_week <- as.factor(bankdata$day\_of\_week) |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| # Split Sample Validation: 80% train / 20% test |
|  |

|  |
| --- |
| set.seed(1) |
|  |

|  |
| --- |
| train\_bank <- sample(1:nrow(bankdata),0.8\*nrow(bankdata)) # using 80% as training data |
|  |

|  |
| --- |
| train <- bankdata[train\_bank,] |
|  |

|  |
| --- |
| test <- bankdata[-train\_bank,] |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| ## Train new RF model without duration var |
|  |

|  |
| --- |
| set.seed(1) |
|  |

|  |
| --- |
| rf.new4 = randomForest(y ~ ., data = train, mtry = 4, ntree = 50, importance = T) |
|  |

|  |
| --- |
| rf.new4 |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| ## Predict classifications on test data  |
|  |

|  |
| --- |
| new.test4 <- predict(rf.new4, test[,1:20], type = "response") |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| # Confusion matrix for RF with m = 4 |
|  |

|  |
| --- |
| rf.new.CF <- table(predicted = new.test4, actual = test$y) |
|  |

|  |
| --- |
| names(rf.new.CF) = paste("Confusion matrix when m = 4:",sep = " ") |
|  |

|  |
| --- |
| rf4\_conf.matrix |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| # Classification Accuracy when m = 4 |
|  |

|  |
| --- |
| new.accuracy4 <- (rf.new.CF[1,1] + rf.new.CF[2,2])/sum(rf.new.CF) |
|  |

|  |
| --- |
| names(new.accuracy4) = paste("Accuracy when m = 4:",sep = " ") |
|  |

|  |
| --- |
| new.accuracy4 |
|  |

|  |
| --- |
| ## 89.0%  |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| # Convert prediction classifications to vote in order for ROC to calculate it as input  |
|  |

|  |
| --- |
| new.test4 <- predict(rf.new4, test[,1:20], type = "vote") |
|  |

|  |
| --- |
| new.df.4 <- as.data.frame(new.test4) |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| # ROC New RF model without duration var  |
|  |

|  |
| --- |
| par(pty = "s") |
|  |

|  |
| --- |
| roc(test$y, new.df.4[,1], plot = TRUE, legacy.axes = TRUE, percent = TRUE,  |
|  |

|  |
| --- |
| xlab = "100 - Specificity % (FPR)", ylab = "Sensitivity % (TPR)", |
|  |

|  |
| --- |
| col = "darkmagenta", lwd = 3, print.auc = TRUE, print.auc.x = 45) |
|  |

|  |
| --- |
| legend("bottomright", legend="New RF with m = 4", col = "darkmagenta", lwd = 3) |
|  |

|  |
| --- |
| roc(test$y, new.test4[,1])  |
|  |

|  |
| --- |
| ## AUC = 79.0% |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| ### -------------------------------------------------------------------------------- ### |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| ## EXTRA CREDIT: h2o package, REQUIRES JAVA TO BE INSTALLED  |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| ### -------------------------------------------------------------------------------- ### |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| ###### EXTRA CREDIT DEEP LEARNING |
|  |

|  |
| --- |
| library(h2o) #requires Java runtime environment  |
|  |

|  |
| --- |
| h2o.init()  |
|  |

|  |
| --- |
| bankdata <- h2o.importFile("bank-additional-full.csv", header = TRUE, sep = ';') |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| bankdata$job <- as.numeric(as.factor(bankdata$job)) |
|  |

|  |
| --- |
| bankdata$marital <- as.numeric(as.factor(bankdata$marital)) |
|  |

|  |
| --- |
| bankdata$education <- as.numeric(as.factor(bankdata$education)) |
|  |

|  |
| --- |
| bankdata$month <- as.numeric(as.factor(bankdata$month)) |
|  |

|  |
| --- |
| bankdata$contact <- as.numeric(as.factor(bankdata$contact)) |
|  |

|  |
| --- |
| bankdata$poutcome <- as.numeric(as.factor(bankdata$poutcome)) |
|  |

|  |
| --- |
| bankdata$day\_of\_week <- as.numeric(as.factor(bankdata$day\_of\_week)) |
|  |

|  |
| --- |
| bankdata$default<- ifelse(bankdata$default == "yes", 1, 0) |
|  |

|  |
| --- |
| bankdata$housing <- ifelse(bankdata$housing== "yes", 1, 0) |
|  |

|  |
| --- |
| bankdata$loan<- ifelse(bankdata$loan== "yes", 1, 0) |
|  |

|  |
| --- |
| bankdata$y <- ifelse(bankdata$y== "yes", 1, 0) |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| h2o.table(bankdata$y) |
|  |

|  |
| --- |
| bankdata$y <- as.factor(bankdata$y) |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| bankh2o <- as.h2o(bankdata) |
|  |

|  |
| --- |
| splits <- h2o.splitFrame(bankh2o, c(0.8,0.19), seed=1) |
|  |

|  |
| --- |
| print(splits[[1]]) |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| ####### FOR IMBALANCED DATA ########### |
|  |

|  |
| --- |
| glm\_result<-h2o.glm(1:20, 21, splits[[1]]) |
|  |

|  |
| --- |
| glm\_perm <- h2o.performance(glm\_result) |
|  |

|  |
| --- |
| glm\_confusionmatrix\_train <- h2o.confusionMatrix(glm\_result) |
|  |

|  |
| --- |
| glm\_confusionmatrix\_train |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| glm\_confusionmatrix\_validation <- h2o.confusionMatrix(glm\_result,splits[[2]]) |
|  |

|  |
| --- |
| glm\_confusionmatrix\_validation |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| glm\_confusionmatrix\_test <- h2o.confusionMatrix(glm\_result,splits[[3]]) |
|  |

|  |
| --- |
| glm\_confusionmatrix\_test |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| ####### GLM FOR BALANCED DATA ###### |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| glm\_result<-h2o.glm(1:20,21, splits[[1]],family = "binomial",balance\_classes = TRUE,nfolds = 5,seed = 1) |
|  |

|  |
| --- |
| glm\_result |
|  |

|  |
| --- |
| predict(glm\_result,splits[[1]]) |
|  |

|  |
| --- |
| glm\_perm <- h2o.performance(glm\_result) |
|  |

|  |
| --- |
| glm\_confusionmatrix\_train <- h2o.confusionMatrix(glm\_result) |
|  |

|  |
| --- |
| glm\_confusionmatrix\_train |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| #predict(glm\_result,splits[[2]]) |
|  |

|  |
| --- |
| glm\_confusionmatrix\_validation <- h2o.confusionMatrix(glm\_result,splits[[2]]) |
|  |

|  |
| --- |
| glm\_confusionmatrix\_validation |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| #predict(glm\_result,splits[[2]]) |
|  |

|  |
| --- |
| glm\_confusionmatrix\_test <- h2o.confusionMatrix(glm\_result,splits[[3]]) |
|  |

|  |
| --- |
| glm\_confusionmatrix\_test |
|  |

|  |
| --- |
| glm\_perm\_test <- h2o.performance(glm\_result,splits[[3]]) |
|  |

|  |
| --- |
| glm\_auc <- h2o.auc(glm\_perm\_test) |
|  |

|  |
| --- |
| print("This is the AUC for GLM in Test set :-") |
|  |

|  |
| --- |
| print(glm\_auc) |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| ### CHECKING THE VARIABLE IMPORTANCE ### |
|  |

|  |
| --- |
| #var\_importance <- h2o.varimp(glm\_result) |
|  |

|  |
| --- |
| #var\_importance |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| ##### RANDOM FOREST FOR BALANCED DATA ##### |
|  |

|  |
| --- |
| randomfr <- h2o.randomForest(1:20,21, splits[[1]],max\_depth = 10,min\_rows = 30,balance\_classes = TRUE,sample\_rate = 0.3,seed = 1) |
|  |

|  |
| --- |
| randomfr |
|  |

|  |
| --- |
| predict(randomfr,splits[[1]]) |
|  |

|  |
| --- |
| rf\_perm <- h2o.performance(randomfr) |
|  |

|  |
| --- |
| randomforest\_confusionmatrix\_train <- h2o.confusionMatrix(randomfr) |
|  |

|  |
| --- |
| randomforest\_confusionmatrix\_train |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| predict(randomfr,splits[[2]]) |
|  |

|  |
| --- |
| randomforest\_confusionmatrix\_validation <- h2o.confusionMatrix(randomfr,splits[[2]]) |
|  |

|  |
| --- |
| randomforest\_confusionmatrix\_validation |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| randomforest\_confusionmatrix\_test <- h2o.confusionMatrix(randomfr,splits[[3]]) |
|  |

|  |
| --- |
| randomforest\_confusionmatrix\_test |
|  |

|  |
| --- |
| rf\_perm\_test <- h2o.performance(randomfr,splits[[3]]) |
|  |

|  |
| --- |
| randomforest\_auc <- h2o.auc(rf\_perm\_test) |
|  |

|  |
| --- |
| print("This is the AUC for Random Forest in Test set :-") |
|  |

|  |
| --- |
| print(randomforest\_auc) |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| ######### DEEP LEARNING MODEL FOR BALANCED DATA ##### |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| dl\_new\_result <- h2o.deeplearning(1:20,21, splits[[1]], activation = "Rectifier",hidden = c(200,200),epochs = 25,balance\_classes = TRUE,variable\_importances = TRUE,shuffle\_training\_data = TRUE,nfolds = 5,seed = 1) |
|  |

|  |
| --- |
| h2o.varimp(dl\_new\_result) |
|  |

|  |
| --- |
| dl\_new\_result |
|  |

|  |
| --- |
| predict(dl\_new\_result,splits[[1]]) |
|  |

|  |
| --- |
| deep\_learning\_perm <- h2o.performance(dl\_new\_result) |
|  |

|  |
| --- |
| deep\_learning\_confusionmatrix\_train <- h2o.confusionMatrix(dl\_new\_result) |
|  |

|  |
| --- |
| deep\_learning\_confusionmatrix\_train |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| predict(dl\_new\_result,splits[[2]]) |
|  |

|  |
| --- |
| deep\_learning\_confusionmatrix\_validation <- h2o.confusionMatrix(dl\_new\_result,splits[[2]]) |
|  |

|  |
| --- |
| deep\_learning\_confusionmatrix\_validation |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| deep\_learning\_confusionmatrix\_test<-h2o.confusionMatrix(dl\_new\_result,splits[[3]]) |
|  |

|  |
| --- |
| deep\_learning\_confusionmatrix\_test |
|  |

|  |
| --- |
| deep\_learning\_perm\_test <- h2o.performance(dl\_new\_result,splits[[3]]) |
|  |

|  |
| --- |
| deep\_learning\_perm\_auc <- h2o.auc(deep\_learning\_perm\_test) |
|  |

|  |
| --- |
| print("This is the AUC for Deep Learning in Test set :-") |
|  |

|  |
| --- |
| print(deep\_learning\_perm\_auc) |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| #var\_plot <- h2o.varimp\_plot(dl\_new\_result) |
|  |

|  |
| --- |
| #var\_plot |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| plot(glm\_perm,type = "roc") |
|  |

|  |
| --- |
| plot(rf\_perm,type = "roc") |
|  |

plot(deep\_learning\_perm,type = "roc")