AIS A THON

2023-04-19

Prompt 3: Build a model to predict whether a particular transaction will have an error.

Solution: Create a descion tree since the y-variable is binary and not necessarily quantitative. Want to predict whether given certain factors might hypothetically lead to a error in a transaction

libraries

library(datasets)  
library(tidyverse)

## ── Attaching packages ─────────────────────────────────────── tidyverse 1.3.2 ──  
## ✔ ggplot2 3.4.0 ✔ purrr 0.3.5   
## ✔ tibble 3.1.8 ✔ dplyr 1.0.10  
## ✔ tidyr 1.2.1 ✔ stringr 1.4.1   
## ✔ readr 2.1.3 ✔ forcats 0.5.2   
## ── Conflicts ────────────────────────────────────────── tidyverse\_conflicts() ──  
## ✖ dplyr::filter() masks stats::filter()  
## ✖ dplyr::lag() masks stats::lag()

library(caret)

## Loading required package: lattice  
##   
## Attaching package: 'caret'  
##   
## The following object is masked from 'package:purrr':  
##   
## lift

library(grid)  
library(gridExtra)

##   
## Attaching package: 'gridExtra'  
##   
## The following object is masked from 'package:dplyr':  
##   
## combine

library(GGally)

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

library(rpart)  
library(rpart.plot)  
library(corrplot)

## corrplot 0.92 loaded

library(ROCR)  
library(readxl)

read dataset

set1 =read.csv("AIS-A-THON\_dataset.csv", header =T, na.strings = "?")  
names(set1)

## [1] "account\_number\_sender" "account\_number\_receiver"   
## [3] "transaction\_timestamp" "transfer\_amount"   
## [5] "beginning\_balance\_sender" "bank\_sender"   
## [7] "os\_sender" "os\_receiver"   
## [9] "unanimous\_agreement" "transaction\_error"

head(set1)

## account\_number\_sender account\_number\_receiver transaction\_timestamp  
## 1 409000362497 409000611074 6/29/2017 12:00  
## 2 409000362498 409000611073 6/29/2017 12:02  
## 3 409000362499 409000611072 6/29/2017 12:04  
## 4 409000362500 409000611071 6/29/2017 12:06  
## 5 409000362501 409000611070 6/29/2017 12:08  
## 6 409000362502 409000611069 6/29/2017 12:10  
## transfer\_amount beginning\_balance\_sender bank\_sender os\_sender os\_receiver  
## 1 13390 100000 BofA Android OS Android OS  
## 2 1800 200000 Wells Fargo Apple iOS Apple iOS  
## 3 500 250 Chase Apple iOS Apple iOS  
## 4 19580 550000 Citizens Android OS Apple iOS  
## 5 8160 600000 Capital One Apple iOS Apple iOS  
## 6 4180 650000 Citi Apple iOS Apple iOS  
## unanimous\_agreement transaction\_error  
## 1 1 1  
## 2 1 0  
## 3 1 1  
## 4 0 1  
## 5 1 0  
## 6 1 0

str(set1)

## 'data.frame': 200 obs. of 10 variables:  
## $ account\_number\_sender : num 4.09e+11 4.09e+11 4.09e+11 4.09e+11 4.09e+11 ...  
## $ account\_number\_receiver : num 4.09e+11 4.09e+11 4.09e+11 4.09e+11 4.09e+11 ...  
## $ transaction\_timestamp : chr "6/29/2017 12:00" "6/29/2017 12:02" "6/29/2017 12:04" "6/29/2017 12:06" ...  
## $ transfer\_amount : num 13390 1800 500 19580 8160 ...  
## $ beginning\_balance\_sender: num 100000 200000 250 550000 600000 650000 700000 750000 800000 850000 ...  
## $ bank\_sender : chr "BofA" "Wells Fargo" "Chase" "Citizens" ...  
## $ os\_sender : chr "Android OS" "Apple iOS" "Apple iOS" "Android OS" ...  
## $ os\_receiver : chr "Android OS" "Apple iOS" "Apple iOS" "Apple iOS" ...  
## $ unanimous\_agreement : int 1 1 1 0 1 1 1 1 0 1 ...  
## $ transaction\_error : int 1 0 1 1 0 0 0 1 1 1 ...

dictionary:

account\_number\_sender, account\_number\_receiver - nominal data fields with account numbers for both parties: senders and receivers of money

transaction\_timestamp - field containing the date and time of the transaction

transfer\_amount - the total amount being transferred from sender’s account to the receiver’s account

beginning\_balance\_sender - this field is to inform if the sender had sufficient funds beforehand

bank\_sender - categorical field containing the bank which stores the sender’s money

os\_sender, os\_receiver - type of operating systems that the sender and receiver have on their phones

unanimous\_agreement - binary field indicating whether both parties agreed to the transaction

transaction\_error - binary field indicating whether the transaction resulted in an error

preprocess data

set2 <- set1[c(4:10)] #get rid of irrelevant features  
#check structure of dataset  
names(set2)

## [1] "transfer\_amount" "beginning\_balance\_sender"  
## [3] "bank\_sender" "os\_sender"   
## [5] "os\_receiver" "unanimous\_agreement"   
## [7] "transaction\_error"

dim(set2)

## [1] 200 7

head(set2)

## transfer\_amount beginning\_balance\_sender bank\_sender os\_sender os\_receiver  
## 1 13390 100000 BofA Android OS Android OS  
## 2 1800 200000 Wells Fargo Apple iOS Apple iOS  
## 3 500 250 Chase Apple iOS Apple iOS  
## 4 19580 550000 Citizens Android OS Apple iOS  
## 5 8160 600000 Capital One Apple iOS Apple iOS  
## 6 4180 650000 Citi Apple iOS Apple iOS  
## unanimous\_agreement transaction\_error  
## 1 1 1  
## 2 1 0  
## 3 1 1  
## 4 0 1  
## 5 1 0  
## 6 1 0

#convert to factor since they are going to be used as classifiers  
set2$transaction\_error <- as.factor(set2$transaction\_error)  
set2$unanimous\_agreement <- as.factor(set2$unanimous\_agreement)  
str(set2)

## 'data.frame': 200 obs. of 7 variables:  
## $ transfer\_amount : num 13390 1800 500 19580 8160 ...  
## $ beginning\_balance\_sender: num 100000 200000 250 550000 600000 650000 700000 750000 800000 850000 ...  
## $ bank\_sender : chr "BofA" "Wells Fargo" "Chase" "Citizens" ...  
## $ os\_sender : chr "Android OS" "Apple iOS" "Apple iOS" "Android OS" ...  
## $ os\_receiver : chr "Android OS" "Apple iOS" "Apple iOS" "Apple iOS" ...  
## $ unanimous\_agreement : Factor w/ 2 levels "0","1": 2 2 2 1 2 2 2 2 1 2 ...  
## $ transaction\_error : Factor w/ 2 levels "0","1": 2 1 2 2 1 1 1 2 2 2 ...

run descion tree model

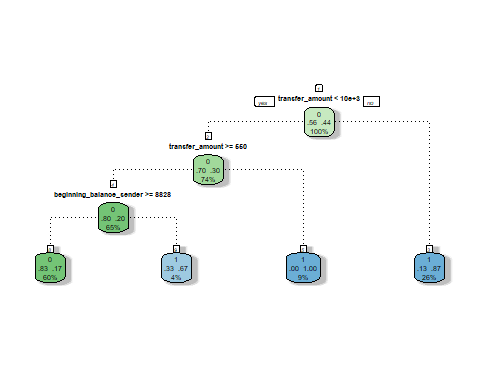
treeFit <- rpart(transaction\_error~.,data=set2, method = 'class')  
print(treeFit)

## n= 200   
##   
## node), split, n, loss, yval, (yprob)  
## \* denotes terminal node  
##   
## 1) root 200 89 0 (0.5550000 0.4450000)   
## 2) transfer\_amount< 10160 148 44 0 (0.7027027 0.2972973)   
## 4) transfer\_amount>=550 130 26 0 (0.8000000 0.2000000)   
## 8) beginning\_balance\_sender>=8828 121 20 0 (0.8347107 0.1652893) \*  
## 9) beginning\_balance\_sender< 8828 9 3 1 (0.3333333 0.6666667) \*  
## 5) transfer\_amount< 550 18 0 1 (0.0000000 1.0000000) \*  
## 3) transfer\_amount>=10160 52 7 1 (0.1346154 0.8653846) \*

#function to create a visual plot  
fig <- function(width, heigth){  
 options(repr.plot.width = width, repr.plot.height = heigth)  
}

visualize the tree

fig(20, 20)  
rattle::fancyRpartPlot(treeFit, type = 1, sub = "")



Train and test data set for accuracy

set.seed(343)  
  
set2[, 'train'] <- ifelse(runif(nrow(set2)) < 0.75, 1, 0)  
  
trainSet <- set2[set2$train == 1,]  
testSet <- set2[set2$train == 0, ]  
  
trainColNum <- grep('train', names(trainSet))  
trainColNum

## [1] 8

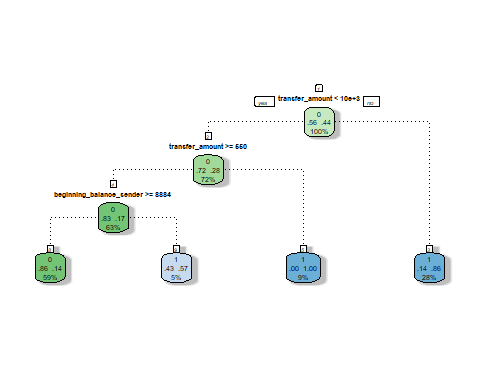
trainSet <- trainSet[, -trainColNum]  
testSet <- testSet[, -trainColNum]

Descision tree for trained set

treeFit <- rpart(transaction\_error~.,data=trainSet,method = 'class')  
print(treeFit)

## n= 155   
##   
## node), split, n, loss, yval, (yprob)  
## \* denotes terminal node  
##   
## 1) root 155 68 0 (0.5612903 0.4387097)   
## 2) transfer\_amount< 10160 112 31 0 (0.7232143 0.2767857)   
## 4) transfer\_amount>=550 98 17 0 (0.8265306 0.1734694)   
## 8) beginning\_balance\_sender>=8884 91 13 0 (0.8571429 0.1428571) \*  
## 9) beginning\_balance\_sender< 8884 7 3 1 (0.4285714 0.5714286) \*  
## 5) transfer\_amount< 550 14 0 1 (0.0000000 1.0000000) \*  
## 3) transfer\_amount>=10160 43 6 1 (0.1395349 0.8604651) \*

rattle::fancyRpartPlot(treeFit, type = 1, sub = "")



check accuracy of the tree

Prediction1 <- predict(treeFit,newdata=testSet,type = 'class')  
confusionMatrix(Prediction1,testSet$transaction\_error)

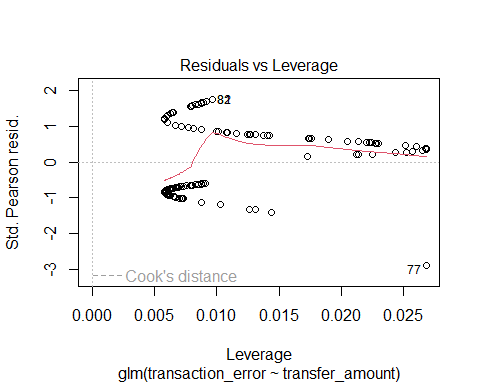
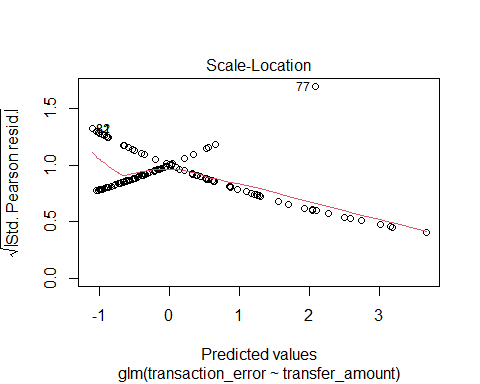
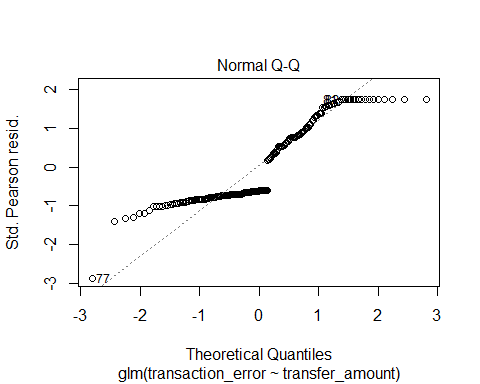
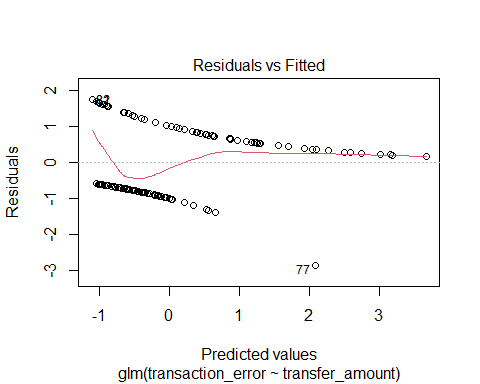
## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 23 7  
## 1 1 14  
##   
## Accuracy : 0.8222   
## 95% CI : (0.6795, 0.92)  
## No Information Rate : 0.5333   
## P-Value [Acc > NIR] : 4.994e-05   
##   
## Kappa : 0.6364   
##   
## Mcnemar's Test P-Value : 0.0771   
##   
## Sensitivity : 0.9583   
## Specificity : 0.6667   
## Pos Pred Value : 0.7667   
## Neg Pred Value : 0.9333   
## Prevalence : 0.5333   
## Detection Rate : 0.5111   
## Detection Prevalence : 0.6667   
## Balanced Accuracy : 0.8125   
##   
## 'Positive' Class : 0   
##

Logistic Regression Model

model <- glm(transaction\_error ~ transfer\_amount ,   
 data = set2, family = binomial)  
summary(model)

##   
## Call:  
## glm(formula = transaction\_error ~ transfer\_amount, family = binomial,   
## data = set2)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -2.0999 -0.9521 -0.8004 1.0220 1.6667   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -1.102e+00 2.273e-01 -4.849 1.24e-06 \*\*\*  
## transfer\_amount 1.158e-04 2.418e-05 4.791 1.66e-06 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 274.83 on 199 degrees of freedom  
## Residual deviance: 242.19 on 198 degrees of freedom  
## AIC: 246.19  
##   
## Number of Fisher Scoring iterations: 4

plot(model)



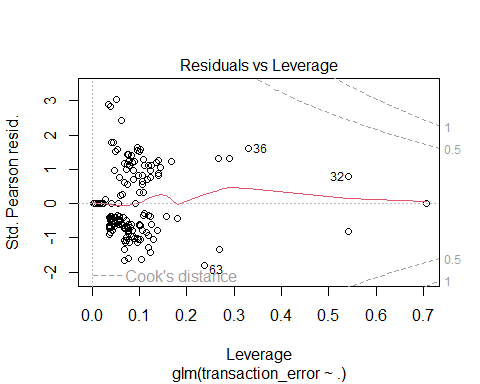
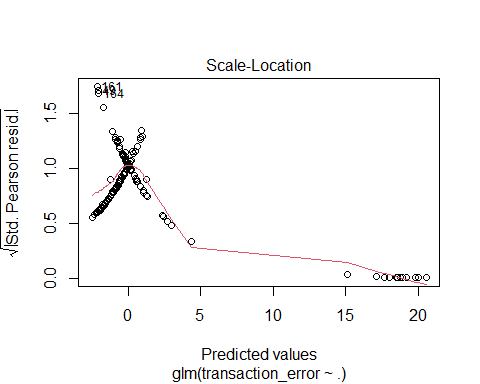
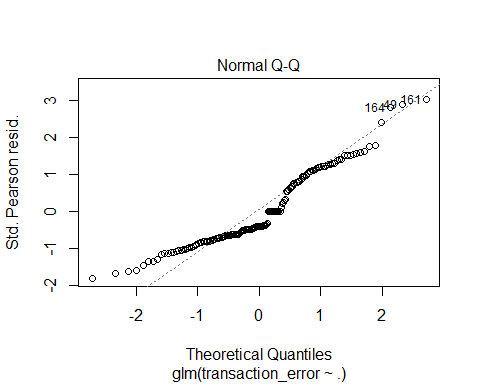
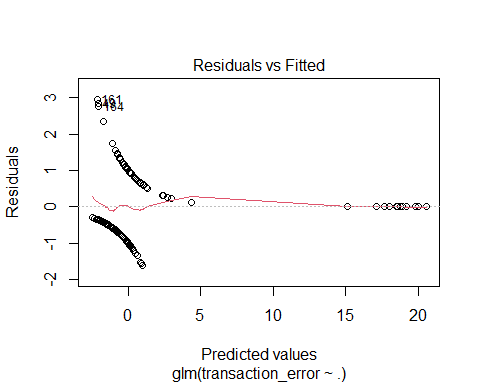
set.seed(343)  
trainIndex <- createDataPartition(set2$transaction\_error, p = .75, list = FALSE,times = 1)  
train <- set2[trainIndex,]  
valid <- set2[-trainIndex,]

Logistic Regression Model

model <- glm(transaction\_error ~. ,   
 data = train, family = binomial)  
summary(model)

##   
## Call:  
## glm(formula = transaction\_error ~ ., family = binomial, data = train)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -1.5961 -0.8704 -0.5153 0.9843 2.1328   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) 1.704e+01 9.788e+02 0.017 0.98611   
## transfer\_amount 1.212e-04 3.934e-05 3.082 0.00206 \*\*  
## beginning\_balance\_sender 2.057e-07 3.327e-07 0.618 0.53640   
## bank\_senderCapital One -1.650e+00 6.984e-01 -2.363 0.01813 \*   
## bank\_senderChase -4.600e-01 6.106e-01 -0.753 0.45120   
## bank\_senderCiti -1.329e+00 6.437e-01 -2.065 0.03892 \*   
## bank\_senderCitizens -2.446e+00 1.243e+00 -1.968 0.04911 \*   
## bank\_senderWells Fargo -9.369e-02 7.312e-01 -0.128 0.89805   
## os\_senderApple iOS -1.466e-01 5.014e-01 -0.292 0.77001   
## os\_senderOther -3.440e-01 1.761e+00 -0.195 0.84507   
## os\_receiverApple iOS 6.907e-02 4.551e-01 0.152 0.87937   
## os\_receiverOther 1.694e+00 1.052e+00 1.610 0.10747   
## unanimous\_agreement1 -1.704e+01 9.788e+02 -0.017 0.98611   
## train -6.377e-01 4.590e-01 -1.389 0.16475   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 207.41 on 150 degrees of freedom  
## Residual deviance: 159.65 on 137 degrees of freedom  
## AIC: 187.65  
##   
## Number of Fisher Scoring iterations: 16

plot(model)



accuracy

predict\_reg <- predict(model, valid, type = 'response')  
y\_predicted <- ifelse(predict\_reg > 0.5, 1, 0)  
y\_predicted <- as.factor(y\_predicted)  
str(y\_predicted)

## Factor w/ 2 levels "0","1": 1 1 2 1 1 1 2 2 1 2 ...  
## - attr(\*, "names")= chr [1:49] "6" "10" "12" "21" ...

table(y\_predicted, valid$transaction\_error)

##   
## y\_predicted 0 1  
## 0 18 10  
## 1 9 12

accuracy <- table(y\_predicted, (valid$transaction\_error))  
sum(diag(accuracy))/sum(accuracy)

## [1] 0.6122449