R Notebook

# Data Processing

library(magrittr)  
library(dplyr)

##   
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':  
##   
## filter, lag

## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

library(Matrix)  
library(tidyverse)

## -- Attaching packages -------------------------------------------- tidyverse 1.3.0 --

## v ggplot2 3.3.0 v purrr 0.3.4  
## v tibble 3.0.1 v stringr 1.4.0  
## v tidyr 1.1.0 v forcats 0.5.0  
## v readr 1.3.1

## -- Conflicts ----------------------------------------------- tidyverse\_conflicts() --  
## x tidyr::expand() masks Matrix::expand()  
## x tidyr::extract() masks magrittr::extract()  
## x dplyr::filter() masks stats::filter()  
## x dplyr::lag() masks stats::lag()  
## x tidyr::pack() masks Matrix::pack()  
## x purrr::set\_names() masks magrittr::set\_names()  
## x tidyr::unpack() masks Matrix::unpack()

library(caret)

## Loading required package: lattice

##   
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':  
##   
## lift

trainData <- read.csv("train.csv")  
testData <- read.csv("test.csv")  
  
# str(test\_Application\_ID)  
  
test\_Application\_ID <- as.character(testData[, 1])  
  
toDefault\_ <- as.integer(factor(trainData[, 52], levels = c("no", "yes"),  
 labels = c("0", "1")))  
str(toDefault\_)  
  
for (i in 1:length(toDefault\_)){  
 if (toDefault\_[i] == "1") {  
 toDefault\_[i] <- 0  
 }else{  
 toDefault\_[i] <- 1  
 }  
}  
# toDefault\_ <- as.factor(toDefault\_)  
head(toDefault\_)  
  
trainData$default\_status <- NULL  
fullData <- rbind(trainData, testData)  
  
fullData <- fullData[, -1]  
  
index\_ <- c(16:20, 35:39, 46:47)  
for (var\_ in index\_) {  
 fullData[, var\_] <- as.factor(fullData[, var\_])  
}  
  
transLog <- c(1, 6:15, 21:32)  
for (ind\_ in transLog) {  
 fullData[, ind\_] <- fullData[, ind\_] ^ (1 / 5)  
}  
  
str(fullData)  
  
#   
# fullData <- fullData %>%   
# mutate(  
# form\_field\_3by33 = log(sqrt(form\_field3 / form\_field33)),  
# form\_field\_1by15by44 = log((form\_field1 / form\_field15) \* form\_field44) ^ (-1),  
# # # form\_field\_42by43 = log(form\_field42 \* form\_field43),  
# form\_field\_1by10by12 = form\_field1 / (form\_field10 \* form\_field12),  
# # form\_field\_8by3 = log(form\_field8 \* form\_field3),  
# # form\_field\_2by50 = sqrt(form\_field2) \* log(form\_field50),  
# form\_field\_1by7 = form\_field1 \* form\_field7,  
# form\_field\_1by30 = form\_field1 \* form\_field30,  
# # form\_field\_26by25 = form\_field26 / form\_field25,  
# form\_field\_13by14 = form\_field13 \* form\_field14,  
# # form\_field\_11by15 = form\_field11 \* form\_field15,  
# # form\_field\_8by13 = form\_field8 / form\_field13,  
# form\_field\_1by25by30 = (form\_field1 \* form\_field25) / form\_field30,  
# form\_field\_1by25by32 = (form\_field1 \* form\_field25) / form\_field32,  
# form\_field\_1by21 = form\_field1 \* form\_field21,  
# # form\_field\_24by21by32 = (form\_field24 \* form\_field21) / form\_field32,  
# # form\_field\_7by10by11 = (form\_field7 \* form\_field10) / form\_field11,  
#   
# ) %>%  
# transform(form\_field2 = sqrt(form\_field2),  
# form\_field50 = log(form\_field50),  
# form\_field32 = log(form\_field32),  
# form\_field7 = form\_field7 ^ (-1))  
#   
  
fullData <- fullData %>%  
 mutate(  
 form\_field\_3by33 = log(sqrt(form\_field3 / form\_field33)),  
 form\_field\_42by43 = log(form\_field42 \* form\_field43),  
 form\_field\_1by15by44 = (form\_field1 / form\_field15) \* form\_field44,  
 form\_field\_1by44 = form\_field1 \* form\_field44,  
 form\_field\_1by2 = log(sqrt(form\_field1 \* form\_field2)),  
 form\_field\_1by3 = log(sqrt(form\_field1 \* form\_field3)),  
 form\_field\_1by6 = log(sqrt(form\_field1 \* form\_field6)),  
 form\_field\_1by7 = log(sqrt(form\_field1 \* form\_field7)),  
 form\_field\_1by9 = log(sqrt(form\_field1 \* form\_field9)),  
 form\_field\_1by10 = log(sqrt(form\_field1 \* form\_field10)),  
 form\_field\_1by11 = log(sqrt(form\_field1 \* form\_field11)),  
 form\_field\_1by22 = log(sqrt(form\_field1 \* form\_field22)),  
 form\_field\_1by23 = log(sqrt(form\_field1 \* form\_field24)),  
 form\_field1div15 = log(form\_field1 / form\_field15),  
 form\_field\_1by15 = form\_field1 \* form\_field15,  
 form\_field\_1by25by30 = (form\_field1 \* form\_field25) / form\_field30,  
 form\_field\_1by25by32 = (form\_field1 \* form\_field25) / form\_field32,  
 form\_field\_1by21by32 = (form\_field1 \* form\_field21) / form\_field32,  
 form\_field\_2by50 = form\_field2 \* form\_field50,  
 form\_field\_2by15by44 = log((form\_field2 / form\_field15) \* form\_field44),  
 form\_field\_2by44 = log(sqrt(form\_field2 \* form\_field44)),  
 form\_field\_2by3 = log(form\_field2 \* form\_field3),  
 form\_field\_2by11 = log(form\_field2 \* form\_field11),  
 form\_field\_2by14 = log(form\_field2 \* form\_field14),  
 form\_field\_2by15 = log(sqrt(form\_field2 / form\_field15)),  
 form\_field\_2by27 = log(sqrt(form\_field2 / form\_field27)),  
 form\_field\_2by28 = log(sqrt(form\_field2 / form\_field28)),  
 form\_field\_2by29 = log(sqrt(form\_field2 / form\_field29)),  
 form\_field\_2by33 = log(sqrt(form\_field2 / form\_field33)),  
 form\_field\_2by34 = log(form\_field2 \* form\_field34),  
 form\_field\_2by30 = log(sqrt(form\_field2 / form\_field30)),  
 form\_field\_2by15 = log(sqrt(form\_field2 \* form\_field15)),  
 form\_field\_2by25by32 = log((form\_field2 \* form\_field25) / form\_field32),  
 form\_field\_2by21by32 = log(sqrt((form\_field2 \* form\_field21) / form\_field32)),  
 form\_field\_26by25 = log(sqrt(form\_field26 \* form\_field25)),  
 form\_field\_8by13 = log(sqrt(form\_field8 \* form\_field13)),  
 form\_field\_24by21by32 = log(sqrt((form\_field24 \* form\_field21) / form\_field32)),  
 form\_field\_7by10by11 = log((form\_field7 \* form\_field10) / form\_field11),  
 # Variables to check out 7, 8, 10, 11, 13, 21, 24, 25, 26, 32  
 )  
# %>%  
# transform(form\_field2 = sqrt(form\_field2),  
# form\_field50 = log(form\_field50),  
# form\_field32 = log(form\_field32))   
# %>%   
 # select(., -c(form\_field5, form\_field12, form\_field17, form\_field20, form\_field39, form\_field50))  
  
str(fullData)  
  
# remove\_feature <- c("form\_field5", "form\_field12", "form\_field17", "form\_field39", "form\_field")  
#   
# fullData <- full  
  
  
  
  
#   
# ohe\_feats = c('form\_field16', 'form\_field17', 'form\_field18', 'form\_field19', 'form\_field20',  
# 'form\_field35', 'form\_field36', 'form\_field37', 'form\_field38', 'form\_field39',  
# 'form\_field46', 'form\_field47')  
# dummies <- dummyVars(~ form\_field16 + form\_field17 + form\_field18 + form\_field19 +form\_field20 +  
# form\_field35 + form\_field36 + form\_field37 + form\_field38 + form\_field39 +  
# form\_field46 + form\_field47, data = fullData)  
# df\_all\_ohe <- as.data.frame(predict(dummies, newdata = fullData))  
# fullData <- cbind(fullData[,-c(which(colnames(fullData) %in% ohe\_feats))],df\_all\_ohe)  
  
# str(fullData)  
# toDefault\_ <- as.integer(toDefault\_)  
  
split\_ <- c(1:nrow(trainData))  
train\_ <- cbind(fullData[split\_, ], toDefault\_)  
test\_ <- fullData[-split\_, ]  
  
# View(train\_)  
  
# ggplot(train\_) +  
# geom\_point(mapping = aes(x = form\_field32, #log(sqrt((form\_field1 \* form\_field44) / form\_field5)),  
# y = log((form\_field42 / form\_field43) \* log(form\_field1)),  
# color = toDefault))  
#   
# ggplot(train\_) +  
# geom\_point(mapping = aes(x = form\_field2,  
# y = form\_field43,  
# color = toDefault\_  
# ))  
#   
# ggplot(train\_) +  
# geom\_point(mapping = aes(x =  
# # log(sqrt((form\_field7 \* form\_field10) / form\_field11)),  
# # log((form\_field7 \* form\_field10) / form\_field11),  
# # (form\_field2 \* form\_field14) ^ (-1),  
# form\_field7 / form\_field8,  
# # (form\_field7 \* form\_field10) / form\_field11,  
# y = form\_field1,  
# color = as.factor(toDefault\_)))  
#   
# ggplot(train\_) +  
# geom\_point(mapping = aes(x = log(form\_field1 / form\_field15),  
# y = form\_field2,  
# color = as.factor(toDefault\_)))  
#   
# ggplot(train\_) +  
# geom\_point(mapping = aes(x = form\_field1 ^ (-1),  
# y = log(form\_field14),  
# color = toDefault))  
#   
# ggplot(train\_) +  
# geom\_point(mapping = aes(x = form\_field1,  
# y = form\_field\_1by15by44,  
# color = toDefault))  
# ggplot(train\_) +  
# geom\_point(mapping = aes(x = form\_field1,  
# y = form\_field14,  
# color = toDefault))

# XGBoost models

# Packages ----------------------------------------------------------------  
  
library(xgboost)

##   
## Attaching package: 'xgboost'

## The following object is masked from 'package:dplyr':  
##   
## slice

library(magrittr)  
library(dplyr)  
library(Matrix)  
library(e1071)  
library(Metrics)

##   
## Attaching package: 'Metrics'

## The following objects are masked from 'package:caret':  
##   
## precision, recall

# Data --------------------------------------------------------------------  
  
split\_ <- c(1:nrow(trainData))  
train\_ <- fullData[split\_, ]  
test\_ <- fullData[-split\_, ]  
  
# str(test\_) # Data whose prediction is to be submitted  
#   
# str(build\_) # 85% of the original train data to be used to build the model  
# str(validate\_) # 15% of the original train data to be used for validation  
  
# defaultIndex <- ncol(train\_)  
  
  
# build\_features <- data.matrix(train\_)  
# build\_label <- factor(build\_[, defaultIndex])  
#   
# params <- list(booster = "gbtree", max.depth = 8, eta = 0.01,   
# objective = "binary:logistic", subsample = 0.8)  
#   
# model\_ <- xgboost(params, data = build\_features, label = build\_label,  
# nround = 150, eval\_metric = "rmse")  
  
#   
# dvalidate <- xgb.DMatrix(data.matrix(validate\_[, -362]), label = validate\_[, 362], missing = NA)  
#   
# dtrain <- xgb.DMatrix(data.matrix(train\_), label = as.factor(toDefault\_), missing = NA)######  
  
str(train\_)  
  
params <- list(booster = "gbtree",   
 objective = "binary:logistic",   
 eta = 0.00005,   
 gamma = 1,  
 max\_depth = 25,   
 min\_child\_weight = 1,   
 subsample = 1,   
 colsample\_bytree = 1)  
#   
# model\_ <- xgb.train(params = params, data = dtrain, nrounds = 291,   
# print\_every\_n = 10, maximize = F , eval\_metric = "auc")  
  
  
  
# machine = xgboost(dtrain, num\_class = 2 , max.depth = 2,   
# eta = 1, nround = 2,nthread = 2,   
# objective = "multi:softprob")  
#   
# df\_metrics <- function(data, level = NULL, model = NULL) {  
# df\_eval = auc(data[, "obs"], data[, "pred"])  
# names(df\_eval) = c("AUC")  
# df\_eval  
# }  
  
#   
# control <- trainControl(method = "cv",  
# number = 5,  
# classProbs = T,  
# # summaryFunction = df\_metrics,  
# )  
#   
# param\_grid <- expand.grid(eta = 0.1,  
# colsample\_bytree = 0.5,  
# max\_depth = 2,  
# nrounds = 100,  
# gamma = 1,  
# min\_child\_weight = 1,  
# subsample = 0.8)  
#   
# modelxgboost <- train(toDefault\_ ~.,   
# data = build\_,  
# method = "xgbTree",  
# trControl = control,  
# tuneGrid = param\_grid,  
# na.action = na.pass,  
# # metric = "AUC",  
# )  
  
# real\_model <- train(toDefault\_ ~.,   
# data = train\_,  
# method = "xgbTree",  
# trControl = control,  
# tuneGrid = param\_grid,  
# na.action = na.pass,  
# # metric = "AUC",  
# )  
  
  
# str(train\_features)  
   
# train\_features <- as.matrix(train\_)  
# train\_label <- factor(train\_[, defaultIndex])  
#######################################################  
# model\_1 <- xgboost(params,   
# data = dtrain,  
# nround = 12,   
# eval\_metric = "rmse")  
#######################################################  
  
  
#------ Validation ------------------------------------------------------------  
  
# test\_prediction <-  
# predict(model\_1, newdata = data.matrix(validate\_[, -the\_index]))  
#   
# head(test\_prediction)  
#   
# auc(validate\_[, the\_index], test\_prediction)  
#   
# #-------------------------------------------------------------------------------  
#   
# submission <- predict(model\_1, newdata = data.matrix(test\_))  
#   
# submission1 <- data.frame(test\_Application\_ID, submission - 1)  
# names(submission1) <- c("Applicant\_ID", "default\_status")  
#   
# head(submission1)  
# # write.csv(submission1, file = "newFile.csv")  
# # write.csv(submission1, file = "newFile2.csv")  
# write.csv(submission1, file = "xgboost\_submission11.csv")  
  
  
  
  
# 0.7905256  
# 0.787411 without dummies  
# 0.8128544 for 100 rounds  
# 0.8179179 for 50 rounds  
# 0.8208659 for 20 rounds  
# 0.8211754 for 10 rounds  
# 0.8223227 for 15 rounds  
# 0.821877 for 17 rounds  
# 0.8185076 for 5 rounds  
  
  
# 0.8231404 (15 rounds after adding two new features)  
# 0.8221882 (12 rounds after adding two new features)  
  
  
# 0.8258275  
# 0.8252653  
# 0.8255938  
# 0.8243203  
# 0.8221103  
# 0.8238826  
# 0.8239109  
  
  
# 0.82613

# Catboost models

library(catboost)  
library(e1071)  
library(Metrics)  
  
split\_ <- c(1:nrow(trainData))  
train\_ <- fullData[split\_, ]  
test\_ <- fullData[-split\_, ]  
  
  
  
  
# str(validate\_)  
  
# head(train\_)  
  
  
train\_pool <- catboost.load\_pool(data = train\_, label = toDefault\_, cat\_features = c(2))  
  
# -------------------------------------------------  
# 0.9433533 validation  
# 0.843069905 zindi  
params <- list(iterations = 2350,  
 learning\_rate = 0.001,  
 depth = 12,  
 # one\_hot\_max\_size = 255,  
 # l2\_leaf\_reg = 3.5,  
 # loss\_function = 'Logloss',  
 # custom\_loss = 'Logloss',  
 # eval\_metric = 'AUC',  
 random\_seed = 55,  
 # bootstrap\_type = "Bayesian",  
 od\_type = 'Iter',  
 metric\_period = 50,  
 od\_wait = 20)  
# -------------------------------------------------  
  
# 0.9683581 validation  
  
# params <- list(iterations = 5050,  
# learning\_rate = 0.005,  
# depth = 11,  
# # one\_hot\_max\_size = 255,  
# # l2\_leaf\_reg = 3.5,  
# # loss\_function = 'Logloss',  
# # custom\_loss = 'Logloss',  
# # eval\_metric = 'AUC',  
# random\_seed = 55,  
# # bootstrap\_type = "Bayesian",  
# od\_type = 'Iter',  
# metric\_period = 50,  
# od\_wait = 20)  
# -------------------------------------------------  
  
# params <- list(iterations = 5050,  
# learning\_rate = 0.002,  
# depth = 10,  
# # one\_hot\_max\_size = 255,  
# # l2\_leaf\_reg = 3.5,  
# # loss\_function = 'Logloss',  
# # custom\_loss = 'Logloss',  
# # eval\_metric = 'AUC',  
# random\_seed = 55,  
# # bootstrap\_type = "Bayesian",  
# od\_type = 'Iter',  
# metric\_period = 50,  
# od\_wait = 20)  
# -------------------------------------------------  
  
# 0.8509104 validation  
# params <- list(iterations = 5050,  
# learning\_rate = 0.0005,  
# depth = 11,  
# # one\_hot\_max\_size = 255,  
# # l2\_leaf\_reg = 3.5,  
# # loss\_function = 'Logloss',  
# # custom\_loss = 'Logloss',  
# # eval\_metric = 'AUC',  
# # random\_seed = 55,  
# # bootstrap\_type = "Bayesian",  
# # od\_type = 'Iter',  
# metric\_period = 50  
# # od\_wait = 20  
# )  
  
  
# params <- list(iterations = 5550,  
# learning\_rate = 0.005,  
# depth = 11,  
# # one\_hot\_max\_size = 255,  
# # l2\_leaf\_reg = 3.5,  
# # loss\_function = 'Logloss',  
# # custom\_loss = 'Logloss',  
# # eval\_metric = 'AUC',  
# random\_seed = 55,  
# # bootstrap\_type = "Bayesian",  
# od\_type = 'Iter',  
# metric\_period = 50,  
# od\_wait = 20)  
  
###################################  
###################################  
###################################  
###################################  
# catboost\_model <- catboost.train(learn\_pool = train\_pool,   
# # validate\_pool,   
# params = params)  
  
#------------------- Validation Area ------------------------------------------  
  
# trainCheck = data.frame(train\_, toDefault\_)  
# smp\_size = floor(0.45 \* nrow(train\_))  
# set.seed(1234)  
#   
# train\_ind = sample(seq\_len(nrow(trainCheck)), size = smp\_size)  
# validate\_ <- trainCheck[train\_ind, ]  
#   
# the\_index <- ncol(validate\_)  
# catboost\_validation <- catboost.predict(model = catboost\_model,   
# pool = catboost.load\_pool(validate\_[, -the\_index]),   
# prediction\_type = "Probability")  
# auc(validate\_[, the\_index], catboost\_validation)  
#   
# #-------------------- Prediction and submission prep --------------------------  
#   
# catboost\_prediction <- catboost.predict(model = catboost\_model,   
# pool = catboost.load\_pool(test\_),   
# prediction\_type = "Probability")  
# head(catboost\_prediction)  
#   
# write.csv(data.frame(test\_Application\_ID, catboost\_prediction), file = "optimal\_cat5.csv")  
#   
#   
# catboost.get\_feature\_importance(model = catboost\_model)  
#####################################  
#####################################  
#####################################  
#####################################

# Model merging

# cat6 <- read.csv("catboost6.csv") #.8429  
# cat7 <- read.csv("catboost7.csv") #.8419  
# cat11 <- read.csv("catboost11.csv") #.8431##################################  
# cat12 <- read.csv("catboost12.csv")   
# cat15 <- read.csv("catboost15.csv")   
# cat18 <- read.csv("catboost18.csv")  
# cat19 <- read.csv("catboost19.csv")  
# cat20 <- read.csv("catboost20.csv")  
# cat\_1 <- read.csv("optimal\_cat1.csv")  
  
###################################################  
# beauti\_merge3 <- read.csv("beautiful\_merge3.csv")  
# newFile <- read.csv("newFile.csv")  
#############################################  
# head(newFile\_pred)  
# merge7 <- read.csv("merge7.csv")   
# merge8 <- read.csv("merge8.csv")   
# merge9 <- read.csv("merge9.csv")   
# merge10 <- read.csv("merge10.csv")   
#   
# merge7\_pred <- merge7$new\_catboost\_prediction  
# merge8\_pred <- merge8$new\_catboost\_prediction  
# merge9\_pred <- merge9$new\_catboost\_prediction  
# merge10\_pred <- merge10$new\_catboost\_prediction  
################################################  
# beautiPred <- beauti\_merge3[, 2]  
# newFile\_pred <- newFile[, 2]  
  
################################################  
#   
# cat6\_pred <- cat6$catboost\_prediction  
# cat7\_pred <- cat7$catboost\_prediction  
# cat11\_pred <- cat11$catboost\_prediction#############################  
# cat12\_pred <- cat12$catboost\_prediction  
# cat15\_pred <- cat15$catboost\_prediction  
# cat18\_pred <- cat18$catboost\_prediction  
# cat19\_pred <- cat19$catboost\_prediction  
# cat20\_pred <- cat20$catboost\_prediction  
# cat\_1\_pred <- cat\_1$catboost\_prediction  
  
# 0.8426 / (0.8429 + 0.8419 + 0.843 + 0.8426)  
  
  
# new\_pred <- (cat6\_pred \* 0.3) + (cat7\_pred \* 0.2) + (cat11\_pred \* 0.2) +   
# (cat15\_pred \* 0.15) + (cat18\_pred \* 0.15)  
  
# combination1 <- (cat6\_pred \* 0.3) + (cat11\_pred \* 0.3) + (cat15\_pred \* 0.2) + (cat19\_pred \* 0.2)  
# combination2 <- (cat7\_pred \* 0.2) + (cat12\_pred \* 0.15) + (cat18\_pred \* 0.35) + (cat20\_pred \* 0.3)  
#   
# new\_merge <- (combination1 \* 0.85) + (combination2 \* 0.15) +  
# 0.8423 + 0.8415 + 0.8425  
# # +   
# (cat19\_pred \* 0.8423) + (cat20\_pred \* 0.8415) + (cat\_1\_pred \* 0.8425)  
#   
# merging\_merge1 <- (merge7\_pred \* .45) + (merge9\_pred \* .55)  
# merging\_merge2 <- (merge8\_pred \* .55) + (merge10\_pred \* .45)  
#   
#   
# tot\_merge <- (merging\_merge1 \* .35) + (merging\_merge2 \* .65)  
  
  
# new\_catboost\_prediction <- (new\_merge \* .45) + (tot\_merge \* .55)  
  
# new\_catboost\_prediction <- (merge8\_pred \* 0.2) + (merge9\_pred \* 0.7) + (merge10\_pred \* 0.1)  
  
# new\_catboost\_prediction <- (cat11\_pred \* 0.6) + (cat15\_pred \* 0.16) + (cat6\_pred \* 0.14) +   
# (cat18\_pred \* 0.1)  
  
# head(new\_catboost\_prediction)  
  
# default\_status <- (beautiPred \* .5) + (newFile\_pred \* .3) + (cat11\_pred \* .4)##############  
  
###################################  
# write.csv(data.frame(test\_Application\_ID, default\_status),  
# file = "bestPreds10.csv")  
#################################  
# write.csv(data.frame(test\_Application\_ID, new\_catboost\_prediction), file = "complex\_merging3.csv")  
# write.csv(data.frame(test\_Application\_ID, new\_catboost\_prediction), file = "merging\_merge6.csv")  
# write.csv(data.frame(test\_Application\_ID, new\_catboost\_prediction), file = "beautiful\_merge3.csv")  
#   
# a <- 0.8419 + 0.8429 + 0.8431  
# 0.8431/a