R Notebook

Note: I had to comment some of the sensitive/powerful parts out so that R would not be trying to execute them before saving the 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_)
```

```r
# 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)
```

```r
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
```