Tidy XGB markup

Cliff Kintanar

12/11/2018

Tidy XGB

This is a fork of Tidy XGB by kxx, which can be located here: https://www.kaggle.com/kailex/tidy-xgb-all-tables-0-796/code

Load data

Let's start off by loading our data

```
## Parsed with column specification:
## cols(
##   SK_ID_BUREAU = col_double(),
##   MONTHS_BALANCE = col_double(),
##   STATUS = col_character()
## )
```

```
## Parsed with column specification:
## cols(
##   SK_ID_CURR = col_double(),
##   SK_ID_BUREAU = col_double(),
##   CREDIT_ACTIVE = col_character(),
##   CREDIT_CURRENCY = col_character(),
##   DAYS_CREDIT = col_double(),
##   CREDIT_DAY_OVERDUE = col_double(),
##   DAYS_CREDIT_ENDDATE = col_double(),
##   DAYS_ENDDATE_FACT = col_double(),
##   AMT_CREDIT_MAX_OVERDUE = col_double(),
##   CNT_CREDIT_PROLONG = col_double(),
##   AMT_CREDIT_SUM = col_double(),
##   AMT_CREDIT_SUM_DEBT = col_double(),
##   AMT_CREDIT_SUM_LIMIT = col_double(),
##   AMT_CREDIT_SUM_OVERDUE = col_double(),
##   CREDIT_TYPE = col_character(),
##   DAYS_CREDIT_UPDATE = col_double(),
##   AMT_ANNUITY = col_double()
## )
```

```
## Parsed with column specification:
## cols(
##   .default = col_double(),
##   NAME_CONTRACT_STATUS = col_character()
## )
```

```
## See spec(...) for full column specifications.
```

```
## Parsed with column specification:
## cols(
##   SK_ID_PREV = col_double(),
##   SK_ID_CURR = col_double(),
```

```
##   NUM_INSTALMENT_VERSION = col_double(),
##   NUM_INSTALMENT_NUMBER = col_double(),
##   DAYS_INSTALMENT = col_double(),
##   DAYS_ENTRY_PAYMENT = col_double(),
##   AMT_INSTALMENT = col_double(),
##   AMT_PAYMENT = col_double()
## )
```

```
## Parsed with column specification:
## cols(
##   SK_ID_PREV = col_double(),
##   SK_ID_CURR = col_double(),
##   MONTHS_BALANCE = col_double(),
##   CNT_INSTALMENT = col_double(),
##   CNT_INSTALMENT_FUTURE = col_double(),
##   NAME_CONTRACT_STATUS = col_character(),
##   SK_DPD = col_double(),
##   SK_DPD_DEF = col_double()
## )
```

```
## Parsed with column specification:
## cols(
##   .default = col_double(),
##   NAME_CONTRACT_TYPE = col_character(),
##   WEEKDAY_APPR_PROCESS_START = col_character(),
##   FLAG_LAST_APPL_PER_CONTRACT = col_character(),
##   NAME_CASH_LOAN_PURPOSE = col_character(),
##   NAME_CONTRACT_STATUS = col_character(),
##   NAME_PAYMENT_TYPE = col_character(),
##   CODE_REJECT_REASON = col_character(),
##   NAME_TYPE_SUITE = col_character(),
##   NAME_CLIENT_TYPE = col_character(),
##   NAME_GOODS_CATEGORY = col_character(),
##   NAME_PORTFOLIO = col_character(),
##   NAME_PRODUCT_TYPE = col_character(),
##   CHANNEL_TYPE = col_character(),
##   NAME_SELLER_INDUSTRY = col_character(),
##   NAME_YIELD_GROUP = col_character(),
##   PRODUCT_COMBINATION = col_character()
## )
```

```
## See spec(...) for full column specifications.
```

```
## Parsed with column specification:
## cols(
##   .default = col_double(),
##   NAME_CONTRACT_TYPE = col_character(),
##   CODE_GENDER = col_character(),
##   FLAG_OWN_CAR = col_character(),
##   FLAG_OWN_REALTY = col_character(),
##   NAME_TYPE_SUITE = col_character(),
##   NAME_INCOME_TYPE = col_character(),
##   NAME_EDUCATION_TYPE = col_character(),
##   NAME_FAMILY_STATUS = col_character(),
##   NAME_HOUSING_TYPE = col_character(),
##   OCCUPATION_TYPE = col_character(),
```

```
##   WEEKDAY_APPR_PROCESS_START = col_character(),
##   ORGANIZATION_TYPE = col_character(),
##   FONDKAPREMONT_MODE = col_character(),
##   HOUSETYPE_MODE = col_character(),
##   WALLSMATERIAL_MODE = col_character(),
##   EMERGENCYSTATE_MODE = col_character()
## )
```

```
## See spec(...) for full column specifications.
```

```
## Parsed with column specification:
## cols(
##   .default = col_double(),
##   NAME_CONTRACT_TYPE = col_character(),
##   CODE_GENDER = col_character(),
##   FLAG_OWN_CAR = col_character(),
##   FLAG_OWN_REALTY = col_character(),
##   NAME_TYPE_SUITE = col_character(),
##   NAME_INCOME_TYPE = col_character(),
##   NAME_EDUCATION_TYPE = col_character(),
##   NAME_FAMILY_STATUS = col_character(),
##   NAME_HOUSING_TYPE = col_character(),
##   OCCUPATION_TYPE = col_character(),
##   WEEKDAY_APPR_PROCESS_START = col_character(),
##   ORGANIZATION_TYPE = col_character(),
##   FONDKAPREMONT_MODE = col_character(),
##   HOUSETYPE_MODE = col_character(),
##   WALLSMATERIAL_MODE = col_character(),
##   EMERGENCYSTATE_MODE = col_character()
## )
```

```
## See spec(...) for full column specifications.
```

Preprocessing

Now let's preprocess the data

```
fn <- funs(mean, sd, min, max, sum, n_distinct, .args = list(na.rm = TRUE))

sum_bbalance <- bbalance %>%
  mutate_if(is.character, funs(factor(.) %>% as.integer)) %>%
  group_by(SK_ID_BUREAU) %>%
  summarise_all(fn)
rm(bbalance); gc()
```

```
##            used   (Mb) gc trigger   (Mb) limit (Mb)  max used   (Mb)
## Ncells   1866665   99.7    5115214  273.2         NA   5115214  273.2
## Vcells 471841359 3599.9  834754402 6368.7      16384 678826631 5179.1
```

```
sum_bureau <- bureau %>%
  left_join(sum_bbalance, by = "SK_ID_BUREAU") %>%
  select(-SK_ID_BUREAU) %>%
  mutate_if(is.character, funs(factor(.) %>% as.integer)) %>%
  group_by(SK_ID_CURR) %>%
  summarise_all(fn)
rm(bureau, sum_bbalance); gc()
```

```
##             used   (Mb) gc trigger   (Mb) limit (Mb)  max used   (Mb)
## Ncells   1872845  100.1    5115214  273.2         NA   5115214  273.2
## Vcells 478077674 3647.5  834754402 6368.7      16384 678826631 5179.1
```

```
sum_cc_balance <- cc_balance %>%
  select(-SK_ID_PREV) %>%
  mutate_if(is.character, funs(factor(.) %>% as.integer)) %>%
  group_by(SK_ID_CURR) %>%
  summarise_all(fn)
rm(cc_balance); gc()
```

```
##             used   (Mb) gc trigger   (Mb) limit (Mb)  max used   (Mb)
## Ncells   1873051  100.1    5115214  273.2         NA   5115214  273.2
## Vcells 401763876 3065.3  834754402 6368.7      16384 678826631 5179.1
```

```
sum_payments <- payments %>%
  select(-SK_ID_PREV) %>%
  mutate(PAYMENT_PERC = AMT_PAYMENT / AMT_INSTALMENT,
         PAYMENT_DIFF = AMT_INSTALMENT - AMT_PAYMENT,
         DPD = DAYS_ENTRY_PAYMENT - DAYS_INSTALMENT,
         DBD = DAYS_INSTALMENT - DAYS_ENTRY_PAYMENT,
         DPD = ifelse(DPD > 0, DPD, 0),
         DBD = ifelse(DBD > 0, DBD, 0)) %>%
  group_by(SK_ID_CURR) %>%
  summarise_all(fn)
rm(payments); gc()
```

```
##             used   (Mb) gc trigger   (Mb) limit (Mb)  max used   (Mb)
## Ncells   1873158  100.1    5115214  273.2         NA   5115214  273.2
## Vcells 311937853 2379.9  834754402 6368.7      16384 828543816 6321.3
```

```
sum_pc_balance <- pc_balance %>%
  select(-SK_ID_PREV) %>%
  mutate_if(is.character, funs(factor(.) %>% as.integer)) %>%
  group_by(SK_ID_CURR) %>%
  summarise_all(fn)
rm(pc_balance); gc()
```

```
##             used   (Mb) gc trigger   (Mb) limit (Mb)  max used   (Mb)
## Ncells   1873182  100.1    5115214  273.2         NA   5115214  273.2
## Vcells 243225028 1855.7  667803521 5095.0      16384 828543816 6321.3
```

```
sum_prev <- prev %>%
  select(-SK_ID_PREV) %>%
  mutate_if(is.character, funs(factor(.) %>% as.integer)) %>%
  mutate(DAYS_FIRST_DRAWING = ifelse(DAYS_FIRST_DRAWING == 365243, NA, DAYS_FIRST_DRAWING),
         DAYS_FIRST_DUE = ifelse(DAYS_FIRST_DUE == 365243, NA, DAYS_FIRST_DUE),
         DAYS_LAST_DUE_1ST_VERSION = ifelse(DAYS_LAST_DUE_1ST_VERSION == 365243, NA, DAYS_LAST_DUE_1ST_
         DAYS_LAST_DUE = ifelse(DAYS_LAST_DUE == 365243, NA, DAYS_LAST_DUE),
         DAYS_TERMINATION = ifelse(DAYS_TERMINATION == 365243, NA, DAYS_TERMINATION),
         APP_CREDIT_PERC = AMT_APPLICATION / AMT_CREDIT) %>%
  group_by(SK_ID_CURR) %>%
  summarise_all(fn)
rm(prev); gc()
```

```
##             used   (Mb) gc trigger   (Mb) limit (Mb)  max used   (Mb)
## Ncells   1873415  100.1    5115214  273.2         NA   5115214  273.2
```

```
## Vcells 246149633 1878.0  667803521 5095.0      16384 828543816 6321.3
```

```
tri <- 1:nrow(tr)
y <- tr$TARGET

tr_te <- tr %>%
  select(-TARGET) %>%
  bind_rows(te) %>%
  left_join(sum_bureau, by = "SK_ID_CURR") %>%
  left_join(sum_cc_balance, by = "SK_ID_CURR") %>%
  left_join(sum_payments, by = "SK_ID_CURR") %>%
  left_join(sum_pc_balance, by = "SK_ID_CURR") %>%
  left_join(sum_prev, by = "SK_ID_CURR") %>%
  select(-SK_ID_CURR) %>%
  mutate_if(is.character, funs(factor(.) %>% as.integer)) %>%
  mutate(na = apply(., 1, function(x) sum(is.na(x))),
         DAYS_EMPLOYED = ifelse(DAYS_EMPLOYED == 365243, NA, DAYS_EMPLOYED),
         DAYS_EMPLOYED_PERC = sqrt(DAYS_EMPLOYED / DAYS_BIRTH),
         INCOME_CREDIT_PERC = AMT_INCOME_TOTAL / AMT_CREDIT,
         INCOME_PER_PERSON = log1p(AMT_INCOME_TOTAL / CNT_FAM_MEMBERS),
         ANNUITY_INCOME_PERC = sqrt(AMT_ANNUITY / (1 + AMT_INCOME_TOTAL)),
         LOAN_INCOME_RATIO = AMT_CREDIT / AMT_INCOME_TOTAL,
         ANNUITY_LENGTH = AMT_CREDIT / AMT_ANNUITY,
         CHILDREN_RATIO = CNT_CHILDREN / CNT_FAM_MEMBERS,
         CREDIT_TO_GOODS_RATIO = AMT_CREDIT / AMT_GOODS_PRICE,
         INC_PER_CHLD = AMT_INCOME_TOTAL / (1 + CNT_CHILDREN),
         SOURCES_PROD = EXT_SOURCE_1 * EXT_SOURCE_2 * EXT_SOURCE_3,
         CAR_TO_BIRTH_RATIO = OWN_CAR_AGE / DAYS_BIRTH,
         CAR_TO_EMPLOY_RATIO = OWN_CAR_AGE / DAYS_EMPLOYED,
         PHONE_TO_BIRTH_RATIO = DAYS_LAST_PHONE_CHANGE / DAYS_BIRTH,
         PHONE_TO_EMPLOY_RATIO = DAYS_LAST_PHONE_CHANGE / DAYS_EMPLOYED)

docs <- str_subset(names(tr), "FLAG_DOC")
live <- str_subset(names(tr), "(?!NFLAG_)(?!FLAG_DOC)(?!_FLAG_)FLAG_")
inc_by_org <- tr_te %>%
  group_by(ORGANIZATION_TYPE) %>%
  summarise(m = median(AMT_INCOME_TOTAL)) %$%
  setNames(as.list(m), ORGANIZATION_TYPE)

rm(tr, te, fn, sum_bureau, sum_cc_balance,
   sum_payments, sum_pc_balance, sum_prev); gc()
```

```
##             used   (Mb) gc trigger   (Mb) limit (Mb)   max used   (Mb)
## Ncells   1876564  100.3    5115214  273.2         NA    5115214  273.2
## Vcells 287929400 2196.8 1143397044 8723.5      16384 1227639638 9366.2
```

```
tr_te %<>%
  mutate(DOC_IND_KURT = apply(tr_te[, docs], 1, moments::kurtosis),
         LIVE_IND_SUM = apply(tr_te[, live], 1, sum),
         NEW_INC_BY_ORG = recode(tr_te$ORGANIZATION_TYPE, !!!inc_by_org),
         NEW_EXT_SOURCES_MEAN = apply(tr_te[, c("EXT_SOURCE_1", "EXT_SOURCE_2", "EXT_SOURCE_3")], 1, mea
         NEW_SCORES_STD = apply(tr_te[, c("EXT_SOURCE_1", "EXT_SOURCE_2", "EXT_SOURCE_3")], 1, sd))%>%
  mutate_all(funs(ifelse(is.nan(.), NA, .))) %>%
  mutate_all(funs(ifelse(is.infinite(.), NA, .))) %>%
  data.matrix()
```

Prepare the data

```
dtest <- xgb.DMatrix(data = tr_te[-tri, ])
tr_te <- tr_te[tri, ]
tri <- caret::createDataPartition(y, p = 0.9, list = F) %>% c()
dtrain <- xgb.DMatrix(data = tr_te[tri, ], label = y[tri])
dval <- xgb.DMatrix(data = tr_te[-tri, ], label = y[-tri])
cols <- colnames(tr_te)

rm(tr_te, y, tri); gc()
```

```
##            used  (Mb) gc trigger   (Mb) limit (Mb)   max used   (Mb)
## Ncells  2048812 109.5    5115214  273.2         NA    5115214  273.2
## Vcells 51327582 391.6  878192929 6700.1      16384 1227639638 9366.2
```

Train the model

```
cat("Training model...\n")
```

```
## Training model...
```

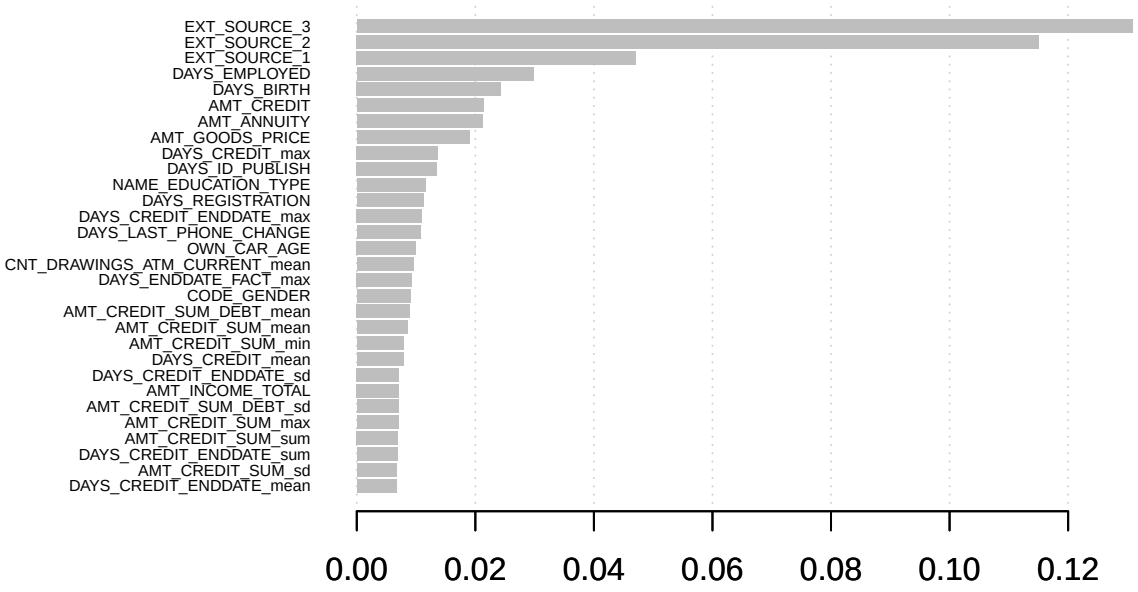
```
p <- list(objective = "binary:logistic",
          booster = "gbtree",
          eval_metric = "auc",
          nthread = 4,
          eta = 0.05,
          max_depth = 6,
          min_child_weight = 30,
          gamma = 0,
          subsample = 0.85,
          colsample_bytree = 0.7,
          colsample_bylevel = 0.632,
          alpha = 0,
          lambda = 0,
          nrounds = 2000)

set.seed(0)
m_xgb <- xgb.train(p, dtrain, p$nrounds, list(val = dval), print_every_n = 50, early_stopping_rounds = 3
```

```
## [1]  val-auc:0.708403
## Will train until val_auc hasn't improved in 300 rounds.
##
## [51] val-auc:0.737537
## [101]    val-auc:0.754598
## [151]    val-auc:0.761548
## [201]    val-auc:0.765373
## [251]    val-auc:0.767363
## [301]    val-auc:0.768654
## [351]    val-auc:0.769488
## [401]    val-auc:0.770097
## [451]    val-auc:0.770456
## [501]    val-auc:0.770823
## [551]    val-auc:0.770880
## [601]    val-auc:0.770717
```

```
## [651]    val-auc:0.770597
## [701]    val-auc:0.770931
## [751]    val-auc:0.771098
## [801]    val-auc:0.770993
## [851]    val-auc:0.770998
## [901]    val-auc:0.771343
## [951]    val-auc:0.771378
## [1001]   val-auc:0.770818
## [1051]   val-auc:0.770651
## [1101]   val-auc:0.770679
## [1151]   val-auc:0.770444
## [1201]   val-auc:0.770087
## Stopping. Best iteration:
## [942]    val-auc:0.771408
```

```
xgb.importance(cols, model=m_xgb) %>%
  xgb.plot.importance(top_n = 30)
```



Format the result and write to file

```
read_csv("./inputs/sample_submission.csv") %>%
  mutate(SK_ID_CURR = as.integer(SK_ID_CURR),
         TARGET = predict(m_xgb, dtest)) %>%
  write_csv(paste0("tidy_xgb_", round(m_xgb$best_score, 5), ".csv"))
```

```
## Parsed with column specification:
## cols(
##   SK_ID_CURR = col_double(),
##   TARGET = col_double()
## )
```