Home Credit Modeling

Rylan Tribush

2024-04-28

Contents

Introduction

Home Credit Group seeks to provide home loans to currently underserved communities, but it only wants to give loans to potential clients with the ability to repay. It is difficult to predict if someone with a limited credit history will be able to repay a loan. Therefore, the overall purpose of this project is to develop a model to predict whether an applicant will repay, based on historical data. The outcome variable is binary, so this is considered a supervised classification problem. We've already done the exploratory data analysis, so this notebook will be exclusively dedicated to predictive modeling. The metric that we will use to judge the models will be AUC. Model interpretablity is not important; we only care about making accurate predictions. After much trial and error, we settled on using the XGBoost model. This notebook will only include the final model and predictions. The data set and the original case competition on which this business problem is based can be found at Kaggle using this link.

Reading the Data

We will start by reading the datasets, which are stored in csv files.

```r
# read the main train and test files
train_data <- read_csv("data_files/application_train.csv", show_col_types = FALSE)
test_data <- read_csv("data_files/application_test.csv", show_col_types = FALSE)
```

```r
# read the supplementary data files
bureau <- read_csv("data_files/bureau.csv", show_col_types = FALSE)
cc_bal <- read_csv("data_files/credit_card_balance.csv", show_col_types = FALSE)
pc_bal <- read_csv("data_files/POS_CASH_balance.csv", show_col_types = FALSE)
prev_app <- read_csv("data_files/previous_application.csv", show_col_types = FALSE)
```

Cleaning

We won't spend a ton of time working with the supplementary data files, but we don't want to ignore them all together. The main approach will be to select the numeric variables, group by customer ID, and then merge on the main data set. When there are multiple rows with the same customer ID, we take the mean of the rows. This approach is almost certainly suboptimal and ignores lots of useful information, but we thought it was better than ignoring the files completely. If we had additional time to work on the project, this would be the first area to improve. After cleaning and merging the data, we drop columns with more than 20% missing data. We then impute the missing values with the median for numeric variables and "Missing" for categorical variables. Finally, we one-hot-encode the data, which is necessary for the XGBoost algorithm.

Warning: some of these code blocks take a while to run!

```r
# select relevant numeric variables, group by customer ID, summarize with mean
bureau <- bureau %>%
  select_if(is.numeric) %>%
  select(-SK_ID_BUREAU) %>%
  group_by(SK_ID_CURR) %>%
  summarise_all(~ mean(.x, na.rm = TRUE))
```

```r
# select relevant numeric variables, group by customer ID, summarize with mean
cc_bal <- cc_bal %>%
  select_if(is.numeric) %>%
  select(-SK_ID_PREV) %>%
  group_by(SK_ID_CURR) %>%
  summarise_all(~ mean(.x, na.rm = TRUE))
```

```r
# select relevant numeric variables, group by customer ID, summarize with mean
pc_bal <- pc_bal %>%
  select_if(is.numeric) %>%
  select(-SK_ID_PREV) %>%
  group_by(SK_ID_CURR) %>%
  summarise_all(~ mean(.x, na.rm = TRUE))
```

```r
# select relevant numeric variables, group by customer ID, summarize with mean
prev_app <- prev_app %>%
  select_if(is.numeric) %>%
  select(-SK_ID_PREV) %>%
  group_by(SK_ID_CURR) %>%
  summarise_all(~ mean(.x, na.rm = TRUE))
```

```r
# separate target variable
train_labels <- train_data$TARGET

# calculate target ratio for later
pos_ratio <- sum(train_labels) / length(train_labels)

# merge and clean data
all_data <- bind_rows(train_data %>% select(-TARGET), test_data) %>%
  left_join(bureau, by = "SK_ID_CURR") %>%
  left_join(cc_bal, by = "SK_ID_CURR") %>%
  left_join(pc_bal, by = "SK_ID_CURR") %>%
  left_join(prev_app, by = "SK_ID_CURR") %>%
  select(-SK_ID_CURR) %>%
  select_if(~mean(is.na(.)) < .2) %>%
  mutate_if(is.numeric, ~ifelse(is.na(.x), median(.x, na.rm = TRUE), .x)) %>%
  mutate_if(is.character, ~ifelse(is.na(.x), "Missing", .x)) %>%
  mutate_if(is.character, as.factor)

# one hot encoding
all_matrix <- data.matrix(one_hot(as.data.table(all_data)))

# split off training data
train_matrix <- all_matrix[1:nrow(train_data),]
```

Split Data for Cross Validation

Here we split the training data 75%/25% into a training fold and testing fold. This is necessary since the actual testing data doesn't have a target variable for us to use and it is impossible to judge model performance. We also have to format the data for the XGBoost algorithm using the DMatrix function.

```
# set seed for reproducability
set.seed(1998)

# split data
inTrain <- createDataPartition(train_labels, p = 0.75, list = FALSE)
tr_features <- train_matrix[inTrain,]
te_features <- train_matrix[-inTrain,]
tr_labels <- train_labels[inTrain]
te_labels <- train_labels[-inTrain]
```

```
# format data for algorithm
dtrain <- xgb.DMatrix(data = tr_features, label = tr_labels)
dtest <- xgb.DMatrix(data = te_features, label = te_labels)
```

Modeling

Here you will see the final model and hyperparameters that we settled on. Please be aware that this is not the only model we tried but the result of many hours of trial and error. If we included every model in this notebook, it would be thousands of lines long.

```
# build model
model_1 <- xgboost(data = dtrain,
                   max_depth = 4,
                   nrounds = 150,
                   scale_pos_weight = pos_ratio,
                   objective = "binary:logistic",
                   eval_metric = "auc")
```

```
# testing AUC score
pred_1 <- predict(model_1, dtest)
auc(roc(te_labels, pred_1, algorithm = 2))
```

```
## Setting levels: control = 0, case = 1

## Setting direction: controls < cases

## Area under the curve: 0.7648
```

Final Model with All Data

Now we repeat the model using all the training data and generate the file of predictions to submit to Kaggle.

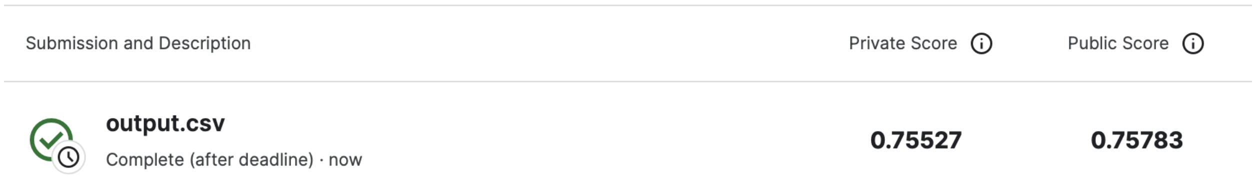
```r
# format data for algorithm
dtrain_f <- xgb.DMatrix(data = all_matrix[1:nrow(train_data),], label = train_labels)
dtest_f <- xgb.DMatrix(data = all_matrix[(nrow(train_data)+1):nrow(all_data),])
```

```r
# build model
model_final <- xgboost(data = dtrain_f,
                       max_depth = 4,
                       nrounds = 150,
                       scale_pos_weight = pos_ratio,
                       objective = "binary:logistic",
                       eval_metric = "auc")
```

```r
# generate predictions and write to csv file
pred_final <- predict(model_final, dtest_f)
read_csv("data_files/sample_submission.csv", show_col_types = FALSE) %>%
  mutate(SK_ID_CURR = as.integer(SK_ID_CURR),
         TARGET = pred_final) %>%
  write_csv("output.csv")
```

Results

Here are the final results from kaggle.



Submission and Description	Private Score	Public Score
output.csv Complete (after deadline) · now	**0.75527**	**0.75783**

Figure 1: submission results