Home Credit Modeling

Whitney Holt

2024-11-25

Contents

Introduction

Home Credit is an international consumer finance provider focused on responsibly lending to people with little to no credit history. To continue serving the unbanked, the company needs to confidently and accurately predict which prospective borrowers are likely to repay loans. Accurate loan repayment predictions enable Home Credit to foster financial inclusion while safeguarding the necessary enterprise profitability to sustain its mission.

The purpose of this project is to create a model to accurately predict which prospective borrowers are likely to repay loans. The specific target variable we will be predicting is called "target", and represents each client's ability to repay a loan (1 - client with payment difficulties: he/she had late payment more than X days on at least one of the first Y installments of the loan in our sample, 0 - all other cases).

The purpose of this modeling notebook is to:

- Set up a training set and a validation set using application_train.csv data set to do cross-validation.
- Identify the performance benchmark established by the majority class classifier.
- Fit a basic logistic regression.
- Explore additional algorithms.
- State performance metrics for the best model: in-sample and estimated out-of-sample AUC, and the resulting Kaggle score & leaderboard ranking.

Loading Packages & Setting up the Workbook

Data Preparation

In the exploratory data analysis (EDA) stage of this project, we …

- Converted categorical variables to factors.
- Imputed missing data.
- Removed near zero variance predictors.
- Removed higly correlated predictors.
- Conducted feature engineering with binning.
- Ensured factor levels were aligned between the train and test sets.

We also decided to use just 5 percent of the available training data to train our models. This 5% needed to be balanced using down sampling.

Feature Engineering

The calculated columns below have been adapted from winning Kaggle notebooks.

```
# Feature Engineering
FivePercBalancedTrain['annuity_income_percentage'] =
  FivePercBalancedTrain['AMT_ANNUITY'] / FivePercBalancedTrain['AMT_INCOME_TOTAL']

FivePercBalancedTrain['credit_to_annuity_ratio'] =
  FivePercBalancedTrain['AMT_CREDIT'] / FivePercBalancedTrain['AMT_ANNUITY']

FivePercBalancedTrain['credit_to_goods_ratio'] =
  FivePercBalancedTrain['AMT_CREDIT'] / FivePercBalancedTrain['AMT_GOODS_PRICE']

FivePercBalancedTrain['credit_to_income_ratio'] =
  FivePercBalancedTrain['AMT_CREDIT'] / FivePercBalancedTrain['AMT_INCOME_TOTAL']

FivePercBalancedTrain['income_credit_percentage'] =
  FivePercBalancedTrain['AMT_INCOME_TOTAL'] / FivePercBalancedTrain['AMT_CREDIT']

FivePercBalancedTrain['income_per_person'] =
  FivePercBalancedTrain['AMT_INCOME_TOTAL'] / FivePercBalancedTrain['CNT_FAM_MEMBERS']

FivePercBalancedTrain['payment_rate'] =
  FivePercBalancedTrain['AMT_ANNUITY'] / FivePercBalancedTrain['AMT_CREDIT']

FivePercBalancedTrain['phone_to_birth_ratio'] =
  FivePercBalancedTrain['DAYS_LAST_PHONE_CHANGE'] / FivePercBalancedTrain['DAYS_BIRTH']
```

Train Test Split

Split the balanced 5 percent into train and test data.

```r
# Set seed for reproducibility
set.seed(123)

# Define values for train set, witholding 30% for the train set
inTrain <- createDataPartition(FivePercBalancedTrain$TARGET, p = 0.7, list=FALSE)

train_set <- FivePercBalancedTrain[inTrain,]
test_set <- FivePercBalancedTrain[-inTrain,]

# Aligning factor levels between the train and test sets
align_factor_levels <- function(train_set, test_set) {
  # Convert all character variables in train and test sets to factors
  train_set[] <- lapply(train_set, function(x) if (is.character(x)) as.factor(x) else x)
  test_set[] <- lapply(test_set, function(x) if (is.character(x)) as.factor(x) else x)

  # Identify all factor variables in the train set
  factor_vars <- names(train_set)[sapply(train_set, is.factor)]

  # Loop through each factor variable
  for (var in factor_vars) {
    # Ensure "Other" level exists in the train set
    train_set[[var]] <- addNA(train_set[[var]])
    levels(train_set[[var]])[is.na(levels(train_set[[var]]))] <- "Other"

    # Convert test set variable to character
    test_set[[var]] <- as.character(test_set[[var]])

    # Replace unseen levels in the test set with "Other"
    test_set[[var]][!test_set[[var]] %in% levels(train_set[[var]])] <- "Other"

    # Convert back to factor and ensure levels match the train set
    test_set[[var]] <- factor(test_set[[var]], levels = levels(train_set[[var]]))
  }

  # Return the updated datasets
  return(list(train_set = train_set, test_set = test_set))
}

# Usage:
aligned_data <- align_factor_levels(train_set, test_set)
train_set <- aligned_data$train_set
test_set <- aligned_data$test_set
```

Modeling Process

Discuss the candidate models you considered, the process for selecting the best model, cross-validation procedures, hyperparameter tuning.

Candidate models:

- Majority Class Classifier
- Simple Logistic Regression
- LASSO Regression
- Naive Bayes

Baseline Model: Majority Class Classifier

```r
# Defining the majority class
majority_class <- names(sort(table(train_set$TARGET), decreasing = TRUE))[1]

# In-sample performance (Train Set)

## Creating a vector of predictions where all are the majority class
majority_class_predictions_train <- rep(majority_class, nrow(train_set))

## Calculating probabilities
majority_class_probabilities_train <- ifelse(
  majority_class_predictions_train == majority_class, 1, 0)

## Calculating AUC for the in-sample predictions
majority_class_roc_curve_train <- roc(train_set$TARGET,
                                      majority_class_probabilities_train)
```

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

## Setting direction: controls < cases
```

```r
majority_class_auc_value_train <- auc(majority_class_roc_curve_train)
print(majority_class_auc_value_train)
```

```
## Area under the curve: 0.5
```

```r
# Estimated Out-of-sample performance (Test Set)

## Creating a vector of predictions where all are the majority class
majority_class_predictions_test <- rep(majority_class, nrow(test_set))

## Calculating probabilities
majority_class_probabilities_test <- ifelse(
  majority_class_predictions_test == majority_class, 1, 0)

## Calculating AUC for the in-sample predictions
majority_class_roc_curve_test <- roc(test_set$TARGET, majority_class_probabilities_test)
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

```r
majority_class_auc_value_test <- auc(majority_class_roc_curve_test)
print(majority_class_auc_value_test)
```

```
## Area under the curve: 0.5
```

The in-sample AUC using a Majority Class Classifier is 0.5.

The estimated out-of-sample AUC using a Majority Class Classifier is 0.5.

Logistic Regression

```r
# Building a simple Logistic Model
Logistic_Model <- glm(TARGET ~ . - SK_ID_CURR, data = train_set, family = binomial)

# In-sample performance (Train Set)

## Generating predicted probabilities on the training set
train_probs <- predict(Logistic_Model, newdata = train_set, type = "response")

## Computing ROC and AUC
train_roc <- roc(train_set$TARGET, train_probs)
```

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

## Setting direction: controls < cases
```

```r
train_auc <- auc(train_roc)

## Printing AUC
print(paste("In-Sample (Train Set) AUC:", train_auc))
```

```
## [1] "In-Sample (Train Set) AUC: 0.773380069839713"
```

```r
# Estimated Out-of-sample performance (Test Set)

## Generating predicted probabilities on the test set
test_probs <- predict(Logistic_Model, newdata = test_set, type = "response")

## Computing ROC and AUC
test_roc <- roc(test_set$TARGET, test_probs)
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

```r
test_auc <- auc(test_roc)

## Printing AUC
print(paste("Out-of-Sample (Test Set) AUC:", test_auc))
```

```
## [1] "Out-of-Sample (Test Set) AUC: 0.659405725853094"
```

```r
table(test_set$TARGET)
```

```
##
##   0   1
## 364 380
```

The in-sample AUC using Logistic Regression is 0.77.

The estimated out-of-sample AUC using Logistic Regression is 0.66.

Naive Bayes Classifier

Basic Naive Bayes model with all predictors, except the ID variable.

```r
# Building a simple Naive Bayes Model
NB_Model <- naiveBayes(TARGET ~ . - SK_ID_CURR, data = train_set)

# In-sample performance (Train Set)

## Generating predicted probabilities on the training set
### Extract probability of the positive class
train_probs <- predict(NB_Model, newdata = train_set, type = "raw")[, 2]

## Computing ROC and AUC
train_roc <- roc(train_set$TARGET, train_probs)
```

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

## Setting direction: controls < cases
```

```r
train_auc <- auc(train_roc)

## Printing the AUC
print(paste("In-Sample (Train Set) AUC:", train_auc))
```

```
## [1] "In-Sample (Train Set) AUC: 0.680056587395721"
```

```r
# Estimated Out-of-sample performance (Test Set)

## Generating predicted probabilities on the test set
### Extract probability of the positive class
test_probs <- predict(NB_Model, newdata = test_set, type = "raw")[, 2]

## Computing ROC and AUC
test_roc <- roc(test_set$TARGET, test_probs)
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

```r
test_auc <- auc(test_roc)

## Printing the AUC
print(paste("Out-of-Sample (Test Set) AUC:", test_auc))
```

```
## [1] "Out-of-Sample (Test Set) AUC: 0.659600925390399"
```

The in-sample AUC using Naive Bayes is 0.68.

The estimated out-of-sample AUC using Naive Bayes is 0.66.

LASSO Regression

```r
# Preparing the data for glmnet
# Converting the data to matrices, excluding 'SK_ID_CURR' and the target variable
x_train <- model.matrix(~ . - SK_ID_CURR - TARGET, data = train_set)[, -1]
y_train <- train_set$TARGET

x_test <- model.matrix(~ . - SK_ID_CURR - TARGET, data = test_set)[, -1]
y_test <- test_set$TARGET

# Fitting a LASSO Regression Model
LASSO_Model <- cv.glmnet(x_train, y_train, family = "binomial", alpha = 1)

# Viewing the optimal lambda
optimal_lambda <- LASSO_Model$lambda.min
print(paste("Optimal Lambda:", optimal_lambda))
```

```
## [1] "Optimal Lambda: 0.0178519144851847"
```

```r
# In-sample performance (Train Set)

## Generating predicted probabilities on the training set
train_probs <- predict(LASSO_Model, newx = x_train, s = "lambda.min", type = "response")

## Computing ROC and AUC
train_roc <- roc(y_train, train_probs)
```

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

## Warning in roc.default(y_train, train_probs): Deprecated use a matrix as
## predictor. Unexpected results may be produced, please pass a numeric vector.

## Setting direction: controls < cases
```

```r
train_auc <- auc(train_roc)

## Printing the AUC
print(paste("In-Sample (Train Set) AUC:", train_auc))
```

```
## [1] "In-Sample (Train Set) AUC: 0.729318764618963"
```

```r
# Estimated Out-of-sample performance (Test Set)

## Generating predicted probabilities on the test set
test_probs <- predict(LASSO_Model, newx = x_test, s = "lambda.min", type = "response")

## Computing ROC and AUC
test_roc <- roc(y_test, test_probs)
```

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

## Warning in roc.default(y_test, test_probs): Deprecated use a matrix as
## predictor. Unexpected results may be produced, please pass a numeric vector.

## Setting direction: controls < cases
```

```r
test_auc <- auc(test_roc)

## Printing the AUC
print(paste("Out-of-Sample (Test Set) AUC:", test_auc))
```

```
## [1] "Out-of-Sample (Test Set) AUC: 0.676033834586466"
```

The in-sample AUC using LASSO Regression is 0.73.

The estimated out-of-sample AUC using LASSO Regression is 0.68.

Model Performance

Individual Model Performance

Performance of the Baseline Model:

- **Majority Class Classifier**
  - The *in-sample* AUC using a Majority Class Classifier is **0.5**.
  - The *estimated out-of-sample* AUC using a **Majority Class Classifier** is **0.5**.

Performance of Additional Fitted Models:

- **Logistic Regression**
  - The *in-sample* AUC using Logistic Regression is **0.77**.
  - The *estimated out-of-sample* AUC using Logistic Regression is **0.66**.
- **Naive Bayes**
  - The *in-sample* AUC using Naive Bayes is **0.68**.
  - The *estimated out-of-sample* AUC using Naive Bayes is **0.66**.
- **LASSO Regression**
  - The *in-sample* AUC using LASSO Regression is **0.72**.
  - The *estimated out-of-sample* AUC using LASSO Regression is **0.68**.

The LASSO Regression Model had the best estimated out-of-sample AUC (0.68).

We will use this model for our Kaggle Submission.

Kaggle Submission

We already applied all of the EDA transformations to the test set (ImputedTest), but now we need to do the Feature Engineering completed above.

```r
# Feature Engineering
ImputedTest['annuity_income_percentage'] =
  ImputedTest['AMT_ANNUITY'] / ImputedTest['AMT_INCOME_TOTAL']

ImputedTest['credit_to_annuity_ratio'] =
  ImputedTest['AMT_CREDIT'] / ImputedTest['AMT_ANNUITY']

ImputedTest['credit_to_goods_ratio'] =
  ImputedTest['AMT_CREDIT'] / ImputedTest['AMT_GOODS_PRICE']

ImputedTest['credit_to_income_ratio'] =
  ImputedTest['AMT_CREDIT'] / ImputedTest['AMT_INCOME_TOTAL']

ImputedTest['income_credit_percentage'] =
  ImputedTest['AMT_INCOME_TOTAL'] / ImputedTest['AMT_CREDIT']

ImputedTest['income_per_person'] =
  ImputedTest['AMT_INCOME_TOTAL'] / ImputedTest['CNT_FAM_MEMBERS']

ImputedTest['payment_rate'] =
  ImputedTest['AMT_ANNUITY'] / ImputedTest['AMT_CREDIT']

ImputedTest['phone_to_birth_ratio'] =
  ImputedTest['DAYS_LAST_PHONE_CHANGE'] / ImputedTest['DAYS_BIRTH']

# Aligning factor levels
train_set_without_target <- subset(train_set, select = -TARGET - SK_ID_CURR)
aligned_data <- align_factor_levels(train_set_without_target, ImputedTest)
ImputedTest <- aligned_data$test_set
```

Preparing the file for Kaggle Submission:

```r
# Preparing the ImputedTest data to be used in a LASSO Regression
x_imputed <- model.matrix(~ . - SK_ID_CURR, data = ImputedTest)[, -1]

# Predicting probabilities using the LASSO model
predicted_probs <- predict(LASSO_Model,
                           newx = x_imputed,
                           s = "lambda.min",
                           type = "response")

# Preparing submission data frame
submission <- data.frame(SK_ID_CURR = as.integer(ImputedTest$SK_ID_CURR),
                         TARGET = as.vector(predicted_probs))

# Saving submission to CSV without scientific notation
options(scipen = 999)  # Turn off scientific notation
write.csv(submission, "submission.csv", row.names = FALSE)
```

```r
print("Submission file created: submission.csv")
```

```
## [1] "Submission file created: submission.csv"
```

Kaggle Score

The LASSO Regression Model attained a Public Kaggle Score of 0.67659.

This out-of-sample AUC is nearly identical to the estimated out-of-sample score attained through cross validation when building the LASSO Model. This indicates that the model was not overfitted to the selected training set.