Home-Credit-Default-Risk

Julian Fong

March 2023

Contents

Abstract

The Home Credit Default Risk Dataset is a credit loan dataset that aims to provide financial knowledge for institutions that lack the ability to provide proper and positive loan services to their clients. The dataset consists of several data files detailing client's historical and current information along with their loans at the time of application. The target in the dataset is a binary variable indicating whether or not the loan has defaulted. The goal in this project is to leverage a machine learning model to classify which loans are likely to default based on the parameters given to us in the data. The methodology of choice includes applying a Xgboost Classifier which employs a tree-based model to classify which loans are more likely to default than others. In the results section, it was determined that the model outperforms the baseline predictions, and has substantial power in predicting whether or not the client will default in the loan.

1 Dataset

The Home Credit Default Risk Dataset contains seven data files and an extra file detailing all of the column variables in each of the data files. All of the data files contains values that are either integers, floats, or strings. The data file train_application.csv contains the majority of each client's information using the unique primary key "SK_ID_CURR". The remaining data files bring in supplementary information for each loan such as previous application info, bureau info, credit card balance information, and other types of details where we can leverage many-to-one joins using these two keys or data aggregation methods. The only other file which contains a primary key is the bureau.csv file where "SK_BUREAU_ID" is used. A relationship chart between the data files is shown in Figure 1.

The train_application.csv contains our target variable "TARGET" which indicates whether or not the client has defaulted on their loan. There are in total 307511 unique clients in dataset along with 282686 non-default claims and 24825 defaulted claims. This means that the dataset is extremely unbalanced, in which the model methodology will need to account for.

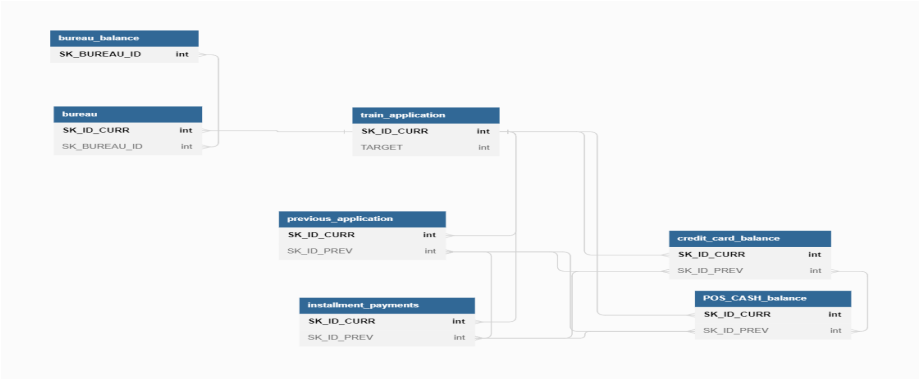
1.1 Data Extraction

All of the data files in .csv format and were downloaded into a local drive. The contents were extracted using the Pandas library for viewing, data selection, and feature engineering.

1.2 Data Selection

After analysis of the predictors in each of the data files, the bureau_balance.csv file was not included in the data analysis as it determined that none of the predic-

Figure 1: Relationship Chart



tors would be useful. Other predictors in the data files such as "FLAG_DOCUMENTS" in train_application.csv were also omitted.

2 Feature Engineering

Since there are multiple data files, feature engineering was performed on the supplementary datasets and then left joined onto the main data file train_application.csv. Afterwards feature engineering was performed again on the aggregated dataset.

2.1 Handling Missing Data

Missing data is heavily prevalent in dataset, with over 47 million missing values in the aggregated dataset and around 90% of rows missing at least one value in the train_application.csv data file. Therefore, many of the missing values remained in the dataset in order to preserve its size. Any column with a majority of missing values over 95% were removed.

2.2 Handling Bad Values and Small Values

Rows that had bad values - values that did not make any sense were removed or replaced. As an example, many rows had values that were classified as 'XNA', and those rows were omitted or the corresponding value replaced as blank in the final dataset.

During the data analysis process, a look at all the possible values there were in each categorical column was taken. If a predictor had a high majority of one single value (over 95%) then the predictor would be removed since the data inside would most likely not improve model prediction.

2.3 Feature Engineering

Various predictors were engineered by taking ratios and averages of other variables. Examples include taking the ratio between the credit amount and the annuity amount of the loan, or taking the average amount of a client's credit card ATM drawings by dividing the total amount withdrawn at the ATM divided by the number of times they drew out of the ATM in a month. Variables inside the original dataset and variables engineered were both included in the final model.

2.4 Data Aggregation

In the supplementary files, the primary key used in train_application.csv is not unique aggregation of the rows with the same primary key was required. Common aggregation techniques include the mean, maximum, sum, or some other or combination of other aggregation methods. Categorical predictors in the supplementary files were converted into one-hot-encoded vectors before the aggregation function as performed. The file name was included inside each new aggregated column name to avoid duplication of column names.

3 Data Analysis

The datatypes in the dataset consists of numerical (float, integer) values along categorical values (strings) describing client and loan data. Some of the numerical predictors were 'counts' containing certain information such as number of family members or 'flags' (0,1) like determining whether or not they owned a mobile. These predictors were converted from numerical to categorical.

3.1 Data Analysis Plots

Density plots were created for numerical predictors and histograms for categorical predictors. The densities and bars were created based on their classification as a defaulted loan or a non-defaulted loan.

Density curves that differ from one another in a numerical column could have potentially significant impact on our predictions when included in the model. For example clients that default on their loan will have a less external source score in Figure 2. In Figure 3, clients are more likely to default than not if they work outside of the city but live inside one.

3.2 Correlation Analysis

Correlation analysis was performed to remove predictors that could contribute to multicollinearity. predictors that had a score of 0.85 or more were removed. After removing the collinear predictors and one-hot-encoding the categorical predictors, the final dataset was saved for the model selection.

Figure 2: Normalized score from external data source

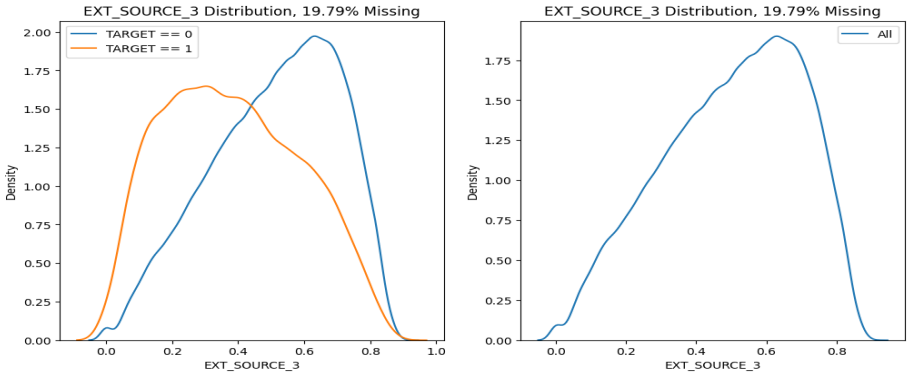
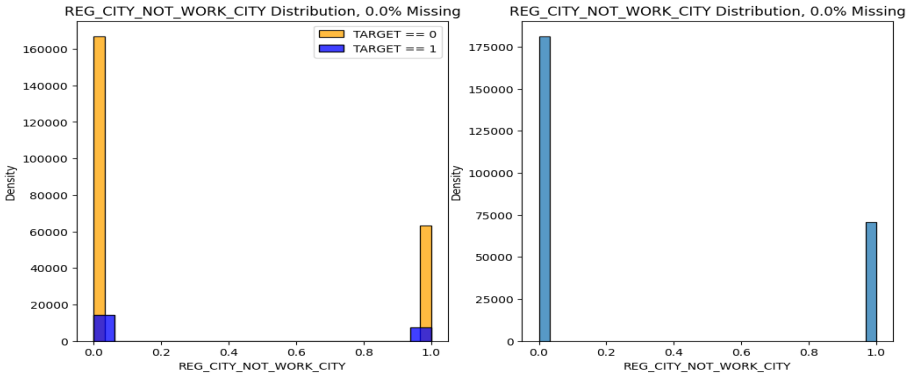


Figure 3: Clients that work outside of the city but live inside the city



4 Model Selection

The final dataset includes 307505 rows and 503 predictors. Having too many predictors can cause the model to overfit and perform poorly on the test set. Instead, feature selection using a XGBoost base model will be used to determine which top 100 predictors to use in our model training process.

4.1 Model Building

The dataset was split into a training set and a test set with a ratio of 70 to 30. The training set and the test set was also ensured to have the same ratio of non-defaults to defaults so there is no bias during training. The base model will include a learning rate of 0.3 and 100 trees will used to train the base model.

4.2 Feature Selection

Using the base model, the amount of predictors will naturally be reduced by selecting 100 of the best performing variables. The method of selecting the top 100 variables is calculated based on the standard Gradient Boosted Tree method, where each predictor is ranked based on how well it improves the performance criteria averaged across all the trees during training. In the base model, the selected predictors include both numerical and categorical, such as days employed, amount in credit, external sources, and so on. Some of the engineered predictors such as average drawings on credit card at ATM were deemed important as well by the base model.

5 Hyperparameter Tuning

Using grid search cross validation, a set of hyperparameters will be used to train a complete XGBoost model. Due to hardware limitations, the number of trees was not a parameter that was experimented with, and was limited to only 100. Increasing the number of trees would increase likely help in increasing model performance but computation time would increase significantly as well.

```
# XGBoost hyperparameter grid
params = {
        'colsample_bytree': [0.5, 1],
        'colsample_bylevel': [0.5, 1],
        'colsample_bynode': [0.5, 1],
        'learning_rate': [0.01, 0.1, 0.2],
        'min_child_weight': [1, 5, 10],
        'subsample': [0.25, 0.5, 1.0],
        'max_depth': [3, 5, 7, 9],
        'scale_pos_weight': [1, 5, 10]
        }
```

6 Results

The XGBoost model employed a binary classification objective, and the best model resulted in this set of hyperparameters.

```
xgb_ = XGBClassifier(n_estimators = 100,
        objective = 'binary:logistic',
        colsample_bylevel = 1,
        colsample_bytree = 0.5,
        colsample_bynode = 0.5,
        learning_rate = 0.1,
        max_depth = 7,
        min_child_weight = 10,
        scale_pos_weight = 1,
        subsample = 1)
```

6.1 Grid Search Cross Validation Results

Grid Search Cross Validation

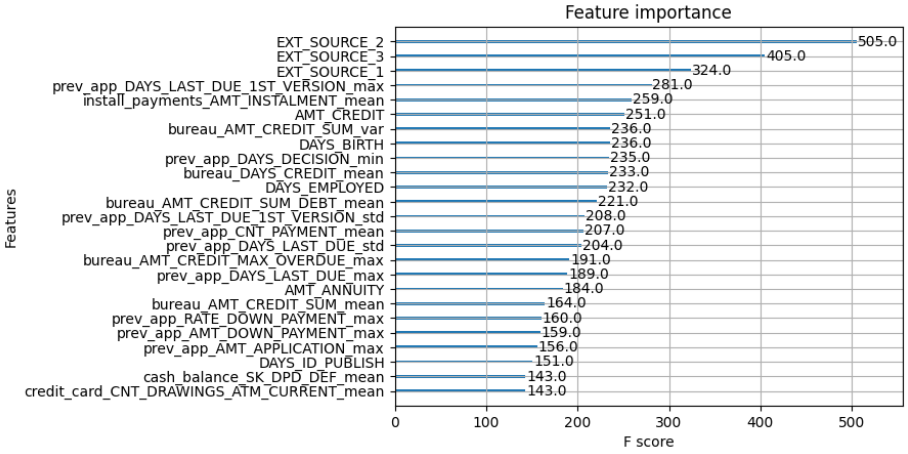
split0_test_score	split1_test_score	split2_test_score	split3_test_score	split4_test_score
0.780575	0.773389	0.777255	0.766353	0.775306
0.780575	0.773389	0.777255	0.766353	0.775306
0.779686	0.772797	0.777770	0.766240	0.772648
0.779317	0.772366	0.777067	0.766968	0.772828
0.778435	0.772989	0.778147	0.764445	0.773973

In total, there were 2,592 possible combinations of model parameters, and 12960 fits were done. The Grid Search Cross Validation table marks the top 5 parameters that resulted in the best test scores. In each of the five cross validations, the test scores are roughly the same, meaning that the model was not overfitted in any of the validation sets.

6.2 Feature Importance

The most important features that impact performance of the model is the external sources. There is reason to believe that may be 3rd party credit scores from a different institution. Other than that, credit card balances, install payments, and previous loan applications all have a smaller but significant impact on whether or not a client will default on a loan.

Figure 4: Feature Importance



6.3 Train and Test Set Results

```
-----------------Train Set Evaluation -----------------
Accuracy Score:  0.9231

Area under ROC Curve:  0.8389

Average Precision Score:  0.4011

Confusion Matrix:
                Positive Prediction  Negative Prediction
Positive Class                197672                  219
Negative Class                 16344                 1018


Classification Report:
              precision    recall  f1-score   support

           0       0.92      1.00      0.96    197891
           1       0.82      0.06      0.11     17362

    accuracy                           0.92    215253
   macro avg       0.87      0.53      0.53    215253
weighted avg       0.92      0.92      0.89    215253
```

```
-----------------Test Set Evaluation -----------------
Accuracy Score:  0.9197

Area under ROC Curve:  0.7738

Average Precision Score:  0.2654

Confusion Matrix:
                Positive Prediction  Negative Prediction
Positive Class                84604                  185
Negative Class                 7226                  237


Classification Report:
              precision    recall  f1-score   support

           0       0.92      1.00      0.96     84789
           1       0.56      0.03      0.06      7463

    accuracy                           0.92     92252
   macro avg       0.74      0.51      0.51     92252
weighted avg       0.89      0.92      0.89     92252
```

The model gives well over 90% accuracy for both the training set and the test set. It is capable of classifying loans that will not default, but the model performance dips when classifying default cases in the test set (value of precision dropping from 0.84 in the training set to 0.54 in the test set).

6.4 Evaluation Metrics

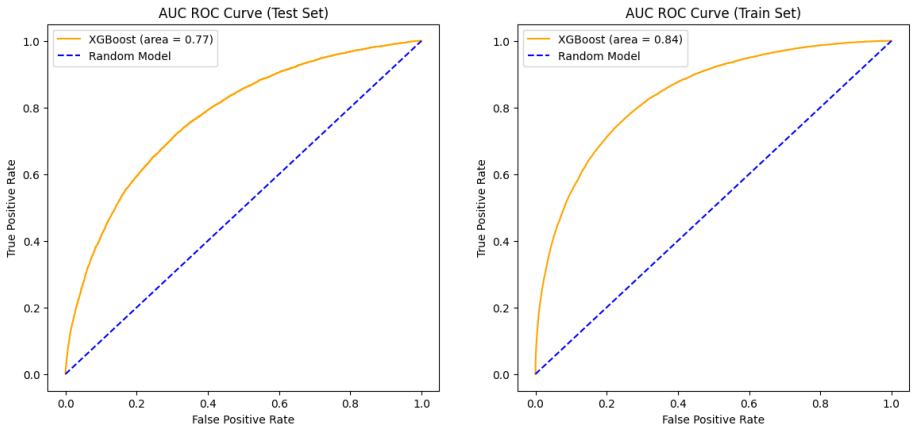
1. ROC Curve

At the end of training, an AUCROC value of 0.77 was obtained for the test set and 0.81 for the training set. This vastly outperforms the random model. The training set AUCROC value and the test set values are quite similar, so there is no overfitting on the model. The high value of AUCROC indicates that the model is capable of distinguishing between default and non-default clients.

2. PR-AUC Curve

The PR-AUC Curve helps identify whether or not the model is capable of classifying positive cases in our dataset. The PRAUC value on the test set is 0.27 and 0.34 on the training set, meaning the model may have slightly overfitted on the training data. The results are still fairly consistent and the model

Figure 5: AUCROC Plot



has the ability to classify positive cases over the random model, which predicts correctly only 8% of the time. The PRAUC value is quite low which can be attributed by the unbalanced nature of the dataset, but is better than the baseline value of 0.08.

3. Lift/Gains Chart

By the 4th decile in Figure 7, the model can correctly predict almost 40% higher than that of the randomly selected samples and by decile 8 it is nearly 100%. This can provide certain cut-off for business purposes. For example, the model can accurately classify over 80% of positive cases using only 60% of the data. Subsequently in the lift chart, the model is outperforming the baseline throughout each decile, with a maximum of 3.5 times the performance right as the beginning of the first decile.

7 Conclusion

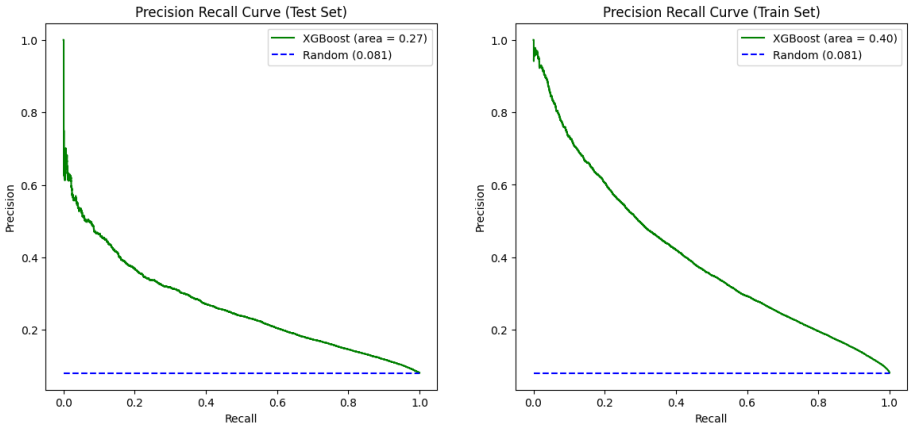
The Home-Credit-Default-Risk Dataset is a large dataset that contains many different predictors. In the model, we were able to improve performance higher than baseline by using a gradient tree-based method. The best model can classify between clients which one is likely to default on their loan.

A Appendix

A.1 Evaluation Metrics

ROC Curve: The AUCROC plot is a graph which describes how well the model is capable of differentiating between classes. A higher AUCROC value

Figure 6: PRAUC Plot



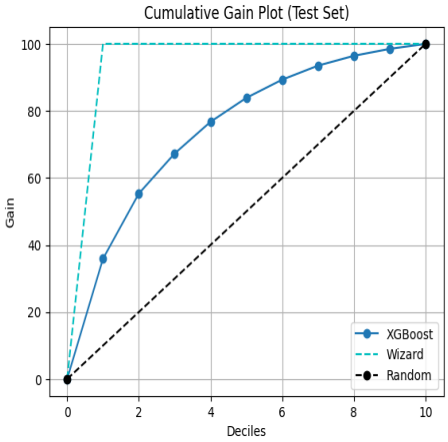


Figure 7: Gains Plot

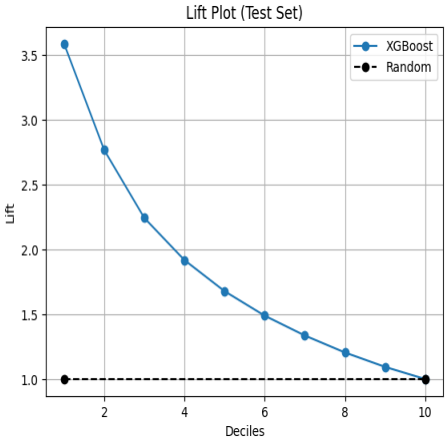


Figure 8: Lift Plot

indicates that one model is better than predicting the correct classes than a model with a lower AUCROC value. A AUCROC value of one indicates that the model has the ability to classify all positive cases as 1 and all negative cases as zero. An AUCROC value of 0.5 indicates that the model has no separation capacity whatsoever.

PR-AUC Curve: The AUCROC plot has a major drawback with unbalanced datasets. We can naturally increase the performance of the model just by classifying all cases as negatives. The PRAUC curve describes how well the model is capable of classifying positive cases, which is an important requirement of a well performing classification model. The PRAUC value

Gains Chart: The Gain Chart demonstrates how powerful the model is when being compared to randomly sampling selected rows. It splits the population in to equally sized deciles, and displays how the performance of the model relative to the random line: the baseline of randomly selected samples.

Lift Chart: The Lift Chart works in tandem with the Gains Chart, plotting how much better the model is than randomly sampled rows in the population. There will be two lines, the baseline which remains at 1 throughout the graph, and a curve demonstrating model performance For example, if the lift chart value for the model is 2.5, then the model outperforms the baseline 2.5 times at the current decile.

A.2 Hyperparameters

Below is a brief overview of each hyperparamater that was used to fine-tune out model:

colsample_bytree: Percentage of features that are randomly sampled to fit each tree.

colsample_bylevel: Fraction of features that we used to train on each level of the tree.

colsample_bynode: Fraction of features that is used to train each node in the tree.

learning_rate: The learning rate is a hyperparameter that we adjust to determine the step size after each iteration of training when minimizing the loss function.

min_child_weight: This is the minimum amount of samples required for a node to be considered for a split. If the weight is too low, we will not use that node anymore for further splitting. This reduces the model complexity and

helps to prevent overfitting.

subsample: By subsampling we are taking a fraction of the training set randomly when training each tree.

max_depth: The maximum depth of a tree when generating splits.

scale_pos_weight: Ratio of positive to negative classes

A.3 Kaggle

The submission.ipynb file is dedicated for submitting the model's performance onto Kaggle. In Kaggle, we obtain the 0.763 public score and a 0.767 private score. The low score may or may not be attributed to the low amount of trees that were used to train the model in each iteration as we only used 100.