MSBD5013 Project 1: Home Credit Default Risk

TSE, Chun Lok
20517188
cltse@connect.ust.hk

YUEN, Zhikun
20505288
zyuen@connect.ust.hk

Abstract

In this project, we have used multiple machine learning methods to predict whether the clients will default or not. We have done some feature selection and engineering so that the AUC score can be further improved. In particular, our LightGBM model [7] achieved the highest AUC score of 0.79236 in the private leaderboard.
GitHub: https://github.com/SamYuen101234/MSBD5013/tree/master/project1

1 Introduction

The competition host Home Credit Group aims to provide loans for clients with insufficient or non-existent credit histories. To investigate and predict which clients will repay their debt, Home Credit Group created this competition and provided this dataset to challenge Kagglers to predict whether their clients are able to repay their debt or not.

2 Data

The dataset contains 8 files. The main application data is in application_train.csv and application_test.csv. In addition, other data such as bureau.csv, bureau_balance.csv, credit_card_balance.csv, installments_payments.csv, POS_CASH_balance.csv and previous_application.csv contains additional information about the clients which might also be useful to identify whether or not they will repay their debts. We have used all of the provided data in this project.

There are two labels, 0 and 1, in the dataset but the dataset is extremely balance. 24,825 are label 1 (client with payment difficulties) and 282,686 are label 0 (other cases).

2.1 Data Preprocessing

The data file is inspected and we identified some abnormalies within the datasets such as:

- 4 people have 'XNA' gender in training data but no client has 'XNA' gender in test data
- extreme outliers in training data
- values that does not make sense according to the data description

These data are either removed, or replaced with null.

For numerical data, we calculated the minimum, maximum, mean, variance and sum of the data aggregated according to SK_ID_CURR.

For categorical data, we first transform them to one hot encoded categories and then group the data according to SK_ID_CURR and calculate the mean value.

Handle Missing Data We handle missing data with different ways for different approaches. For Random Forest and Neural Network, we replace the missing data with the median of the features because the implementation cannot deal with null value. For LightGBM, the implementation can handle missing data, so we keep the null value after aggregating all dataframes together. Features with entire column filled with null data are dropped as they provide no useful information.

Data Normalization After we process the data, some of the data have extreme range of values. In machine learning, normalization is an important step in data pre-processing to deal with features with different extreme range of values. Especially for neural network, data without normalization affects the speed of optimization because the weight update is unequally on different directions. To normalize the data, we use a Min-Max Normalization to re-scale the range of features to [0,1].

2.2 Feature Engineering

Special features are also crafted from the original features to provide additional insight for the model to achieve even higher accuracy. These features are crafted from the point of view of a Home Credit financial analyst and therefore should help increasing the model accuracy. Most of these domain specific features are being discussed in multiple Kaggle discussion [1; 2; 3; 4; 5; 6]. We provide our understanding towards these features below.

For application_train.csv and application_test.csv:

DAYS_EMPLOYED_RATIO = DAYS_EMPLOYED / DAYS_BIRTH

A client that is more recently employed may indicate that his income source is not stable and thus more likely to default.

INCOME_CREDIT_RATIO = AMT_INCOME_TOTAL / AMT_CREDIT

A client that has a high income to credit ratio maybe less likely to default.

INCOME_PER_PERSON = AMT_INCOME_TOTAL / CNT_FAM_MEMBERS

A client with low income per person may indicate a financial hardship and thus more likely to default.

ANNUITY_INCOME_RATIO = AMT_ANNUITY / AMT_INCOME_TOTAL

A client with high annuity to income ratio may indicate most of the money earned is used to repay the debt and thus maybe more likely to default.

PAYMENT_RATE = AMT_ANNUITY / AMT_CREDIT

A client with high payment rate may indicate the client is paying more than the minimum and thus maybe less likely to default.

GOOD_PRICE_CREDIT_RATIO = AMT_GOODS_PRICE / AMT_CREDIT

A client with high goods price to credit ratio may indicate the client is not borrowing a lot of money compared to his total wealth and thus maybe less likely to default.

For credit_card_balance.csv:

LIMIT_USE = AMT_BALANCE / AMT_CREDIT_LIMIT_ACTUAL

A client with high amount used from limit maybe more likely to default.

PAYMENT_DIV_MIN = AMT_PAYMENT_CURRENT / AMT_INST_MIN_REGULARITY

A client with low current payment to minimum payment ratio may indicate financial hardship and thus more likely to default.

LATE_PAYMENT = SK_DPD > 0

A client with late payment maybe more likely to default.

DRAWING_LIMIT_RATIO = AMT_DRAWINGS_ATM_CURRENT / AMT_CREDIT_LIMIT_ACTUAL

A client with high drawing to limit ratio may indicate financial hardship.

For installments_payments.csv:

PAYMENT_RATIO = AMT_PAYMENT / AMT_INSTALMENT
PAYMENT_DIFF = AMT_INSTALMENT - AMT_PAYMENT

A client with low payment to instalment ratio and smaller difference may indicate the client is paying less interest and thus maybe less likely to default.

DAYS_PAST_DUE = (DAYS_ENTRY_PAYMENT - DAYS_INSTALMENT) <= 0
DAYS_BEFORE_DUE = (DAYS_INSTALMENT - DAYS_ENTRY_PAYMENT) <= 0

A client that have past due may indicate financial hardship and thus maybe more likely to default.

For previous_application.csv:

APPLICATION_CREDIT_DIFF = AMT_APPLICATION - AMT_CREDIT
APPLICATION_CREDIT_RATIO = AMT_APPLICATION / AMT_CREDIT

The amount requested and amount received may indicate the trust of the lender towards the client.

2.3 Memory Reduction

After we aggregate all the tables and perform feature engineering, we obtain a super large sparse matrix with shape $307,507 \times 795$. This sparse matrix occupies large amount of memory and makes the later computation and experiments inefficient. To reduce the memory of this sparse matrix, we reduce the precision of the integer and float because some of the values may not need higher precision e.g. int16, int32 or float32. The larger the precision, the more memory is used. Finally, the memory usage is decreased by 72.1% after precision reduction and the later experiments become faster and more efficient.

2.4 Feature Selection

Although we obtain 795 features by aggregating all the tables together, most of the features are useless and noisy. Selecting the most useful features can reduce noise and make training process efficient. To filter out the useless and noisy features, we use Uniform Manifold Approximation and Projection (UMAP) and a modified version of Forward Stepwise Selection with LightGBM to select the most important features.

Uniform Manifold Approximation and Projection (UMAP)

Principle Component Analysis (PCA) is a common technique to reduce data dimensions. PCA finds the directions of maximum variance in high-dimensional data and projects it onto a new subspace with equal or fewer dimensions than the original one. However, since PCA is a linear projection technique, it is unable to capture non-linear dependencies. In this project, we try another non-linear dimension reduction approach, UMAP, to see whether the performance can still be kept if the data is reduced to a low dimension non-linearly. We create a pipline with a UMAP followed by a default LightGBM and use 10-fold GridSearchCV to fine tune the best dimension with the lowest AUC score. However, the average validation AUC is 0.538 only for this pipeline. We believe that the data after dimension reduction by UMAP lose the original representation and information. Also, the non-linear transformation also makes the features become less interpretable. Thus, we will not use UMAP for feature selection in our later experiments.

Forward Stepwise Selection with LightGBM

Forward stepwise selection approach is a method to select features starting from a null model and adding one feature each time. Comparing the result with previous iteration and select features that obtain the highest AUC score. However, we have 795 features and more than 300,000 data. Adding one feature each time makes the selection process become time consuming.

Fortunately, LightGBM, a tree-based approach with multiple boosted trees, provides a way to compute the feature importance (Figure 6) according to the numbers of times the feature is used in a model. Therefore, we modify the forward stepwise selection method (Figure 1). Instead of adding one feature each time, we check the feature importance with LightGBM when we aggregate a new dataframe to the training data, we set a threshold to keep the top $k\%$ of the most important features only and keep the AUC remain unchanged or improve it. Also, removing features with little to none importance from the lists of features can speed up model training and reduce the noisy features. We obtain 0.79x AUC score in validation set with all 795 features. After using this approach, we find that we can keep the AUC at 0.79x with only the top 40% of the most important features. A total of 318 features are left after data selection and feature engineering. We will use these 318 features in our later experiments for different approaches.

3 Methods and Experiments

In this project, we have experimented different machine learning models to evaluate the dataset. The dataset contains both numeric and categorical features. **From this observation, we give a hypothesis that tree-based approaches are more appropriate.** Since the dataset contains more than 300,000 rows in application_train.csv, support vector machine (SVM) is not an efficient approach for such kind of large dataset, which is one of the disadvantage of SVM. In the following experiments, we will focus on comparing the results of Random Forest, LightGBM and Multilayer Perceptron.

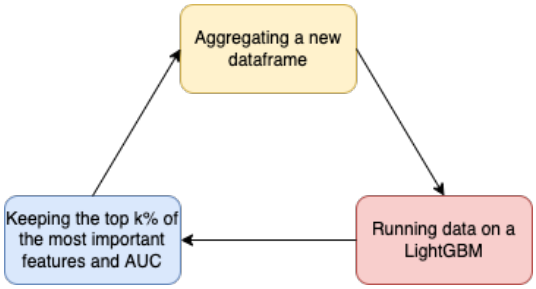

Figure 1: A modified version of forward stepwise selection with LightGBM's feature importance

Table 1: A 3-layer perceptron

layer	size-in	size-out	activation	p
fc1	795	512	ReLU	-
dropout1	512	512	-	0.1
fc2	512	128	ReLU	-
dropout2	128	128	-	0.1
fc3	128	32	ReLU	-
dropout3	32	32	-	0.1
output	128	1	Sigmoid	-

3.1 Random Forest

Random Forest is an improvement of decision tree using bootstrap and random subset of features at splitting to reduce the high variance problem in decision tree. Multiple decision trees are created using a sub-sampling of data with replacement and the trees is split with a subset of features only. Tree-based approaches are good at data with both numeric and categorical features, so we believe that random forest is one of the suitable models for the data.

3.2 LightGBM

LightGBM is a gradient boosting framework that uses tree based learning algorithms. Boosting allows many weak learners to combine together to form a strong learner. The weak learners are created sequentially with the goal of classifying correctly what the previous learner misclassifies. It is especially useful in dealing with extreme data imbalance such as this dataset.

3.3 Multilayer Perceptron (Neural Network)

Neural Network is another powerful tool for classification. According to the universal approximation theorem, a neural network with 1 single hidden layer can approximate any continuous functions. In our project, we build a 3-layer preceptron with 512, 128 and 32 hidden states, as shown in Table 1. We use ReLU as the activation function after each fully-connected layer, Adam as the optimizer, learning rate is 1e-3 and binary cross-entropy loss as the objective function. To prevent overfitting problem, we use both L2 regularisation with λ = 1e-4 and Dropout with dropout rate = 0.1.

3.4 K-Fold Cross-Validation & Early Stopping

In machine learning, how to avoid high variance and overfitting is a very important topic. Overfitting occurs when models fit the noise in data samples as well. This problem is more serious in tree-based approaches and multilayer perceptron. One approach to prevent overfitting is to use validation approach. K-fold cross-validation is one of the most common validation technique to check occurrence of overfitting. In k-fold cross-validation, the whole dataset is split into k folds. Each time we pick k-1 folds as training set and 1 fold as validation set. In the next iteration, we pick the next fold as validation set until all folds are taken as validation set. The final validation loss and AUC are the average of all the validation results of the k iterations. This can increase the reliability and reduce the effect of unbalance sampling. With early stopping, we can stop the training process when the validation loss does not decrease any more after some steps to prevent the overfitting.

In our project, 10-fold cross-validation is used to evaluate our models to ensure that the

Table 2: AUC of 10-fold cross-validation of Random Forest, LightGBM and Multilayer Preceptron

Strain	RF		LGBM		MLP	
	Train	Valid	Train	Valid	Train	Valid
$Fold_1$	0.858	0.753	**0.945**	**0.795**	0.781	0.767
$Fold_2$	0.859	0.748	**0.945**	**0.790**	0.781	0.761
$Fold_3$	0.858	0.743	**0.945**	**0.792**	0.775	0.763
$Fold_4$	0.857	0.751	**0.945**	**0.785**	0.777	0.768
$Fold_5$	0.859	0.743	**0.945**	**0.791**	0.777	0.764
$Fold_6$	0.857	0.742	**0.945**	**0.792**	0.769	0.764
$Fold_7$	0.859	0.751	**0.945**	**0.796**	0.775	0.768
$Fold_8$	0.858	0.748	**0.946**	**0.792**	0.777	0.768
$Fold_9$	0.857	0.749	**0.945**	**0.805**	0.778	0.767
$Fold_{10}$	0.858	0.747	**0.945**	**0.789**	0.772	0.765
Full	0.858	0.748	**0.945**	**0.793**	0.776	0.766

model is really able to generalize well. This gives an average AUC score of our model. To compute the test prediction, we average the prediction value of all the 10 models in the 10-fold CV to perform a model ensembling. Early stopping of 50 rounds is used such that the LightGBM training procedure stops when there are no improvements on validation AUC within the 50 rounds such that the LightGBM does not overfit to the training data significantly. In multilayer preceptron (MLP), we stop the training procedure when there are no improvements on validation loss in 3 epochs.

3.5 Grid Search

Grid search is used to identify the optimal hyperparameters for our models. This enables the model to achieve highest accuracy possible with the dataset given. We used GridSearchCV in Scikit Learn to search for the best set of hyperparameters with the highest AUC score in 10-fold CV. We have done grid search on the some hyperparameters for Random Forest and LightGBM.

3.5.1 Random Forest

Best Hyperparameters From Searching: *{max_depth=13, min_samples_leaf=40, min_samples_split=2}*
The average validation AUC of this best set of hyperparameters is 0.74843.

3.5.2 LightGBM

Best Hyperparameters From Searching: *{max_depth=10, num_leaves=70, min_data_in_leaf=15, min_child_weight=0.0001, bagging_fraction=0.8, bagging_frequency=5, feature_fraction=0.78}*
The average validation AUC of this best set of hyperparameters is 0.79287.

4 Result Analysis

4.1 Performance

The 10-fold cross-validation results of Random Forest, LightGBM, and Multilayer Perceptron are shown in Table 2. From the 10-fold cross-validation of LightGBM, we achieved the highest 0.79287 AUC with all the training data, preprocessing and feature engineering among all the three methods. To visualize the performance of LightGBM, we used the 10 validation sets in the 10-fold cross-validation to plot the mean ROC curve in Figure 2. Although the tree-based approach, LightGBM, achieves the highest AUC score in validation, it is obvious that the results of both tree-based approaches, Random Forest and LightGBM, have higher variance than the 3-layer preceptron. Even though Random Forest and LightGBM use bagging/bootstrap data and boosting, overfitting problem still occurs more seriously than neural network. This also shows that neural network with dropout and L2 regularization can prevent overfitting better in this dataset.

4.2 Loan Repayment

The prediction output of the model is a floating point from [0,1]. If we want to predict whether the credit department should approve the loan for the applicants, we need to set a threshold for prediction. Since the labels are extremely imbalance, if we use 0.5 as the threshold for prediction, the class-specific performance:

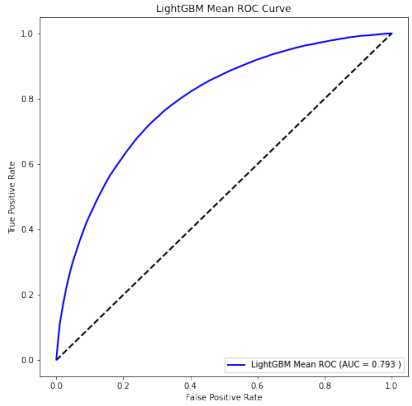

Figure 2: LightGBM Mean ROC Curve for 10-fold validation sets

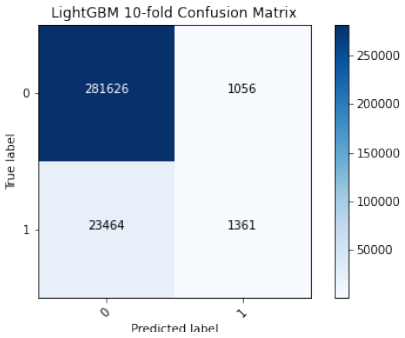

Figure 3: LightGBM Total Confusion Matrix with threshold = 0.5 for all the 10-fold validation sets

- Error rate of client with payment difficulties (label 1): 94.52%
- Sensitivity (TPR, recall of label 1): $1 - 95.52\% = 5.48\%$
- Error rate of client with other cases (label 0): 0.37%
- Specificity: $1 - 0.37\% = 99.63\%$

The confusion matrix is shown in Figure 3. In this case, although most of the other cases (label 0) are classified correctly. Majority of the clients with payment difficulties (label 1) are classified incorrectly. This may affect the credit department's decision and approving lots of risky applications could potentially incur significant losses because many of its borrowers cannot repay the loan. It is obvious that threshold = 0.5 is not a good threshold for final classification.

To help banks reducing the risk, we need to classify clients with payment difficulties (label 1) more accurately but keep the potential clients (label 0) at the same time. We plot the error rate of label 1 and label 0 on the same plot in Figure 4. From the plot, we find the interception of error rate of the two labels is 0.07. Thus, we use 0.07 as the threshold for prediction. The corresponding confusion is shown in Figure 5.

Class-specific performance:

- Error rate of client with payment difficulties (label 1): 27.03%
- Sensitivity (TPR, recall of label 1): $1 - 27.03\% = 72.97\%$
- Error rate of client with other cases (label 0): 28.83%
- Specificity: $1 - 28.83\% = 71.17\%$

The error rate of client with payment difficulties drops from 94.52% to 27.03% and improve the sensitivity to 72.97%. We also keep the error rate and specificity of label 0 in a relatively acceptable level. This can help the bank reduce the risk and keep the potential clients.

4.3 Feature Importance

We display the average feature importance in Figure 6. From the visualization, we indeed observe that some of the engineered features are really important in identifying whether a client will default or not such as PAYMENT_RATE, AMT_ANNUITY, GOOD_PRICE_CREDIT_RATIO and ANNUITY_INCOME_RATIO. This shows that leveraging domain knowledge indeed produce a better result.

Neural Network is less interpretable than Random Forest and LightGBM. However, Explainable AI (XAI) becomes an important field to uncover the black box of neural network. We also try to find the feature importance of a tabular dataset in the 3-layer preceptron following the approach provided by a XAI package [8], in Equation 1. For each feature, we randomly order it again

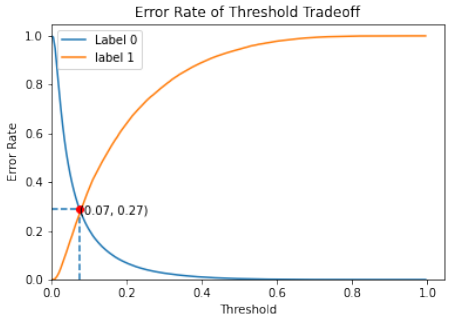

Figure 4: Tradeoff between prediction threshold and error rate

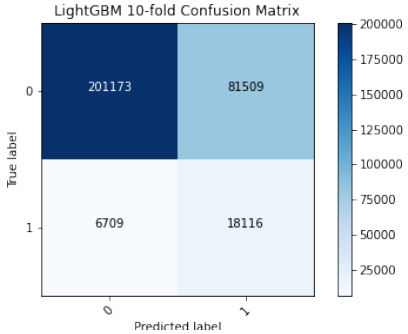

Figure 5: LightGBM Total Confusion Matrix with threshold = 0.07 for all the 10-fold validation sets

and compute the difference of the AUC score between the original features and the features with a random order in k-th column. The more positive difference shows that the feature is more significant. The top-50 most important features of the MLP shares 16 common features with the top-50 most important features of LightGBM. The two features PAYMENT_RATE and AMT_ANNUITY from feature engineering are the common top-50 important features in both MLP and LightGBM. This shows our work of feature engineering can really find some meaningful features again.

$$\frac{1}{N}\sum_{i=0}^{N}[AUC(f(x),y) - AUC(f(x_k),y)]; \forall k = 1,2,\ldots,|x| \tag{1}$$

, where N is the number of iterations, x is all features, x_k is the features that the k-th feature is ordered randomly, $|x|$ is the number of features, y is the ground truth and f is the MLP.

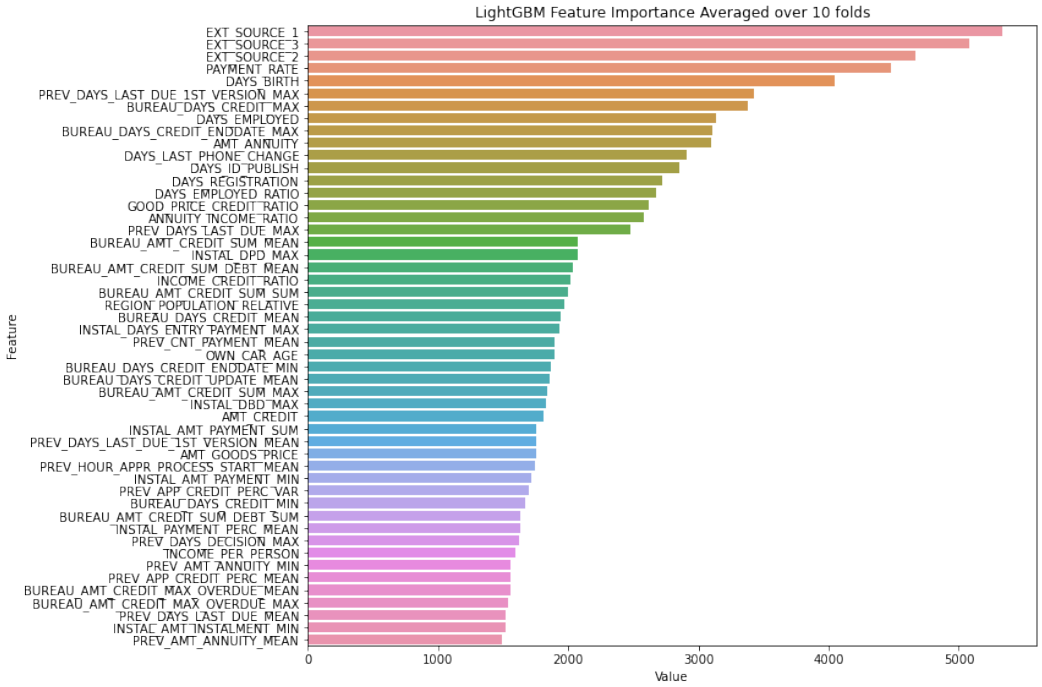

Figure 6: Top 50 most important features selected by LightGBM

5 Conclusion

After performing data cleaning, feature engineering and experiments, we choose the best models, LightGBM, in our 10-fold cross-validation to predict the target of test result. We obtain the best result with AUC = 0.79236 in private score and AUC = 0.79442 in public score in this competition (Figure 7). **This proves our hypothesis that tree-based approaches can usually achieve really good results in tabular data containing both numeric and categorical features.** On the other hand, MLP with suitable fine tuning, e.g. dropout + regularization, can reduce high variance in this dataset.

6 Contribution

TSE, Chun Lok:
Data Preprocessing, Data Cleaning, Feature Engineering, LightGBM training and testing, Report writing.
YUEN, Zhikun:
Data Preprocessing, Memory Reduction, Feature Selection, LightGBM hyperparameters finetuning, Random Forest and Multilayer Perceptron training and testing, Report writing.

7 Kaggle Contest Score

The final predictions are generated using the 10-fold cross-validation LightGBM models. Each of the sub-models are used to generate $1/10$ of the predictions and summing all the predictions together to get the final submission. This is better than only using one single LightGBM model as it averages over all trained models to avoid overfitting. Our final private score on Kaggle is 0.79236 as shown in Figure 7.

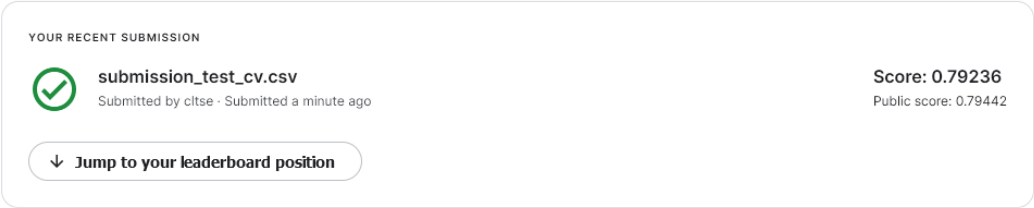

Figure 7: Submission private score on Kaggle

References

[1] Aguiar. LightGBM with Simple Features. Retrieved March 17, 2022, from https://www.kaggle.com/jsaguiar/lightgbm-with-simple-features. 2

[2] Aguiar. LightGBM 7th place solution. Retrieved March 17, 2022, from https://www.kaggle.com/jsaguiar/lightgbm-7th-place-solution. 2

[3] Alijs. 3rd place solution. Retrieved March 17, 2022, from https://www.kaggle.com/c/home-credit-default-risk/discussion/64596. 2

[4] Maxwell. 2nd place solution (team ikiri_DS). Retrieved March 17, 2022, from https://www.kaggle.com/c/home-credit-default-risk/discussion/64722. 2

[5] Tuananhkk. More domain knowledge from former Home Credit analyst. Retrieved March 17, 2022, from https://www.kaggle.com/c/home-credit-default-risk/discussion/63032. 2

[6] Tunguz, B. 1st Place Solution. Retrieved March 17, 2022, from https://www.kaggle.com/c/home-credit-default-risk/discussion/64821. 2

[7] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, Tie-Yan Liu; LightGBM: A Highly Efficient Gradient Boosting Decision Tree; NIPS 2017 1

[8] XAI - An eXplainability toolbox for machine learning; https://github.com/EthicalML/xai 6