Credit Risk Scoring Analysis Based on Machine Learning Models

Ziyue Qiu*, Yuming Li*†, Pin Ni*† and Gangmin Li*‡
*Research Lab for Knowledge and Wisdom, Xi'an Jiaotong-Liverpool University, China
†Department of Computer Science, University of Liverpool, UK
Email: ‡Gangmin.Li@xjtlu.edu.cn

***Abstract*—In the big data era, institutions can easily access a massive number of data describing different aspects of a user. Therefore, credit scoring models are now building from both the past credit records of the applicant, and other personal information including working years and characteristics of owned properties. A wide variety of usable information has required models to extract more expressive features from data and apply the effective models to fit the features. This paper reports our efforts in using feature engineering techniques and machine learning models for credit scoring modeling. Based on the Kaggle Home Credit Default Risk dataset, several current feature engineering techniques and machine learning models have been tested and compared in terms of the AUC score. The results have shown that the LightGBM model training on expert knowledge generated datasets can achieve the best result (About 78% AUC score).**

***Index Terms*—Credit Scoring, Feature Engineering, Machine Learning**

I. Introduction

Credit Scoring is a statistical model for evaluating the repayment ability of loan applicants. Loan applicant's credit information is extensively used in the modeling process. However, since applicants' past credit information is not always available, other potential credit-related data, including property ownership information and family status, is also employed. The potential credit-related data can generally be directly transformed into weak features, but these features are not as expressive as strong features like credit information. To solve this problem and obtain better model performance, two approaches are proposed: one is to enlarge data size used for modeling, the other is to develop more advanced skills to extract effective information from data. Our paper experiment several combinations of feature engineering and machine learning approaches, hoping to propose effective models for credit scoring modeling tasks, especially tasks based on high-dimensional, weakly correlated sparse datasets. Based on the evaluations, our final proposed model achieves 78% of the AUC score on Kaggle Home Credit Default Risk dataset[1].

II. Literature Review

With the growing maturity of the banking market in developing countries. Banks are facing competition not only from other domestic banks but also from mature foreign banks. Credit scores are generated based on a statistical analysis of individual credit reports, and the credit score model is the most successful example of applying statistical models in financial institutions [1]. Large credit bureaus such as Experian keep records of individuals' borrowing and repayment activities to prevent negative credit events, such as mortgage defaults or bankruptcies, from adversely affecting credit companies. Therefore, it is essential to develop a well-functioning credit score model [2]. Existing commonly used methods for constructing credit score models include linear discriminant analysis [3], [4], deep learning methods [5]–[7], classification and regression trees [8], [9], Logistic Regression [10], [11]. In addition to this, Chi et al. [12] selected important variables through the genetic algorithm (GA) to combine the internal behavior scoring model of the bank with the external credit investigation agency scoring model to build a dual scoring model for credit risk management of mortgage accounts. Atiya et al [2] proposed the application of neural networks (NNs) in the bankruptcy prediction of credit risk, which was significantly improved. Altman et al [3] compared neural networks with LDA and concluded that LDA was superior to NN. However, they recommend the integration of the two, as the integration results in significant performance enhancement of the neural network. Khandani et al [13] used machine learning techniques to construct a non-linear, nonparametric prediction model of consumer credit risk. This study used a sample of customers of a major commercial bank, and the model was very accurate in predicting credit events 3 to 12 months in advance.

III. Methodology

(Major steps of the experiment is demonstrated in Fig. 1)

A. Data Pre-processing

1) Anomalies and contradiction detection: To reduce the influence of anomaly samples, we conduct simple examinations over the data. We calculate the age and years of service of each sample from features 'days of birth' and 'days of employed', and drop samples which are negative or greater than 36500.

2) Missing Data Imputation: How to analyze sparse data is a common problem in financial modeling scenarios. We discard features that contain over 70% percent of null values and imply two methods of missing data imputation: mean imputation, medium imputation, and Random Forest imputation.

[1]https://www.kaggle.com/c/home-credit-default-risk

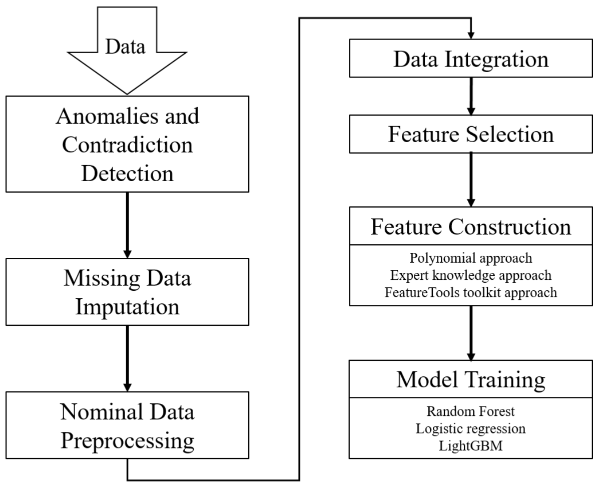

Fig. 1. Major Steps of Experiment

3) Nominal Data Pre-processing: Since nominal data cannot be used as input for most machine learning models, we converted them to numerical data. We achieve these through two approaches, one-hot encoding for features with more categories, and label encoding for features with less than two categories to reduce the dimension of data.

4) Data Integration: To perform a comprehensive analysis of user behavior and make credible predictions on repayment abilities, we merge data from different sources for training.

B. Feature Engineering

1) Feature Selection: To shorten training time and simply final model, we perform feature selection, which is deleting irrelevant and redundant features. To filter out irrelevant features, for each feature, we calculate how many categories its values fall into. If the number of categories is higher than 85% of the total count of samples, we consider this feature to be irrelevant and remove it. Redundant features refer to several mutually correlated features contained in one dataset. We develop a function to calculate the Pearson Correlation Coefficient (PCC) between each pair of features.

$$\rho_{X,Y} = \frac{\operatorname{cov}(X,Y)}{\sigma_X \sigma_Y} \tag{1}$$

We cluster features whose mutual PCC is higher than 75% percent. From each cluster, we choose one feature whose PCC is highest against the target feature and save these features for the further training process.

2) Feature Construction: One of the challenges of modeling credit risk score from user behavior data is that the correlation between target feature and other features are usually weak. By calculating and plotting the PCC of each feature against the target feature, we find out the most correlated features. To improve model performance, we hope to construct stronger features from them. We implement three approaches to construct new features, including the polynomial approach, automatic approach using FeatureTools Toolkit [14] and the expert crafted approach. We saved these constructed features into three different files according to how they are constructed.

C. Model Training

1) Logistic Regression: The specific method of Logistic Regression is as follows: find a suitable hypothesis h, which is a function that needs to be classified to predict the judgment result of the input data. A cost function (loss function) is then constructed, which represents the deviation between the predicted output h and the training data category y, which can be different between h and y $(h - y)$ or other forms between the two. Considering the ”loss” of all training data, the Cost is summed or averaged and recorded as a $J(\theta)$ function, indicating the deviation of the predicted values of all training data from the actual category. The smaller the value of the $J(\theta)$ function, the more accurate the prediction function (i.e. the more accurate the h function), and the minimum value of the $J(\theta)$ function can be found by the Gradient Descent method.

2) Random Forest: Random Forest is an integrated algorithm based on Decision Tree. It is a machine learning method that obtains the final prediction results using voting or averaging after combining multiple independent decision trees, and it is often more accurate and stable than a single tree. The superior performance of a Random Forest compared to a decision tree depends mainly on the random sampling of samples and features and integration algorithms. The former gives it a more stable resistance to overfitting, while the latter gives it a higher accuracy. As a special bagging method, Random Forest uses the Decision Tree as a model in bagging. The specific process is as follows: First, the bootstrap method is used to generate m training sets. Then, a decision tree is constructed for each training set. When the node finds features to split, not all features can be found the maximum index for all features (e.g. The Information Gain), but a part of the features are randomly extracted from the features, find the optimal solution and applied to the node split. Due to the bagging, the method of Random Forest is equivalent to sampling both the sample and the feature, so over-fitting can be avoided.

3) Light GBM: Due to the particularity of the data, there are two problems in selecting the classification model. First, to help the subsequent correlation analysis, the classification model needs to be interpretable and achieve the ordering of the importance of the features. Second, the original data contains a certain amount of noise. Classification models need to have strong generalization capabilities. Considering synthetically, the GBDT (Gradient Boosting Decision Tree) is selected as the classifier model in our method. When GBDT is used as a regression, a new decision tree will be fitted in each round. When GBDT is used as a regression, a new decision tree will be fitted in each round. When GDBT is to fit the t^{th} decision tree, it uses the residual of the regression values of the previous $t - 1$ decision trees on all samples as the value to be fitted. When GBDT is used for classification, the model will fit K Decision Trees at the same time in each round, corresponding to K categories. The Softmax value of the regression value f_m^{t-1} generated by the GBDT model during the $t-$round fitting using the sample in the first $t-1$ round indicates the probability that the sample belongs to the classification, as shown in equation :

$$P_m^t\left(x_i\right)=\frac{\mathrm{e}^{f_m^{\mathrm{t}-1}\left(x_i\right)}}{\sum_{p=1}^K f_p^{t-1}\left(x_i\right)} \tag{2}$$

Let the real label of the sample X_i be represented as a one-hot vector $[q_{m,i}|m=1,2,\cdots,K]$. Only when the sample belongs to the k^{th} class,$X_i$is 1, then the residual of the GBDT model at the m^{th} decision tree in the t^{th} round is $\sum_{i=1}^N q_{m,i}-P_m^t\left(x_i\right)$ Although GBDT performs well in classification on many datasets, it still has two problems: one is that when the model faces a large dataset or with high feature dimensions, the efficiency and scalability of the algorithm is difficult to satisfy. The reason is that the model needs to perform feature selection and node splitting in the process of fitting each Decision Tree. This requires the model to pre-sort the eigenvalues, traverse all possible divided points, and then calculate the information gain to select the optimized split point. The model traverses the entire training set at each fit, which is both computationally intensive and time-consuming. Another problem is that the model does not perform well for an unbalanced training sample classification, so the efficient GBDT—LightGBM is chosen. The model is mainly optimized based on GBDT: on the one hand, the GOSS algorithm is used to optimize the sampling method of sample points during model training; on the other hand, the EFB is used to compress the feature dimension when selecting split points. These two optimization methods can make the model reduce the amount of calculation when selecting the splitting point and improve the performance of the algorithm. GOSS optimization means that when the model splits node for a feature j, the model does not find the optimal threshold d for the sample traversal according to the principle of maximum information gain. The model first sorts the samples according to the gradient values, then selects the $a\%$ samples with the largest gradient value, and randomly extracts b samples from the remaining $(1-a)\%$ of the samples to form a new training set to fit the classifier, and gives the b samples with smaller gradients are multiplied by a larger coefficient. The amount of information gain calculation when splitting a node can be expressed as:

$$\widetilde{V}_j(d)=\frac{1}{n}\left(\frac{\left(\sum_{x_i\in A_l} g_i+\frac{1-a}{b}\sum_{x_i\in B_l} g_i\right)^2}{n_l^j(d)}+\frac{\left(\sum_{x_i\in A_r} g_i+\frac{1-a}{b}\sum_{x_i\in B_r} g_i\right)^2}{n_r^j(d)}\right) \tag{3}$$

Which,

$A_l=\{x_i\in A:x_{ij}\le d\}$, $A_r=\{x_r\in A:x_{ij}>d\}$, $B_l=\{x_i\in B:x_{ij}\le d\}$, $B_r=\{x_r\in B:x_{ij}>d\}$ In this way, each time the Decision Tree is fitted, the model uses non-full data training, which can reduce the amount of calculation from the training data dimension, increase the diversity of each Decision Tree, and improve the generalization ability. Among them, LightGBM uses the Histogram algorithm, the idea is to discrete continuous-floating-point features into k discrete values and construct a Histogram of width k. The training data is then traversed and the cumulative statistic of each discrete value in the histogram is counted. When performing feature selection, it is only necessary to traverse the optimal segmentation point according to the discrete values of the Histogram. The Histogram algorithm has low memory consumption, does not need to store pre-sorted results, and can only save the discretized values. For sparse high-dimensional data, LightGBM can bind mutually exclusive features through the EFB algorithm to achieve the effect of reducing features. This means that the complexity of the Histogram creation will be reduced from $O(data\times feature)$ to $O(data\times bundle)$, thus accelerating the training process of LightGBM.

Therefore, here we will feed the data after data pre-processing into the model, and sort the feature importance by LightGBM, and discard the less relevant data.

IV. EXPERIMENT

A. Experiment Environment

Our experimental environment is as follows: CPU Intel Core i7-7700HQ, RAM: 2.80GHz.

B. Dataset Description

Our experiments are conducted on the Kaggle Home Credit Default Risk dataset. This dataset consists of seven CSV files containing different aspects of user data. We only used three of them, which are the main table of client-provided information at application, a file of clients' previous credits provided by Credit Bureau and a file of clients' previous POS and cash loans with Home Credit. The main table contained 307511 samples of 122 features. We list out the basic information of some features in Table I.

V. RESULT AND ANALYSIS

We derive four datasets from Kaggle Home Credit Default Risk dataset, including one original dataset and three constructed datasets. Three constructed datasets are obtained through adopting different feature engineering approaches, which are polynomial approach, automatic approach, and manually crafted approach. To evaluate the performance improvement brought by feature engineering and model selection, we train three models – Logistic Regression, Random Forest and LightGBM on the four datasets . Table II has illustrated the models' performance on the testing dataset in terms of AUC. According to the experiments, the performance of LightGBM model outperforms the Logistic Regression model and Random Forest model on all datasets generated through different feature engineering processes. Through looking at each column of Table II, the following results are concluded.

For the Random Forest model, the model performs best when trained on the original dataset. The generated features did not improve the model's performance and even decline it. From Table III, we can see that the performance of Random

TABLE I
BASIC INFORMATION OF SAMPLE FEATURES

Feature name	Missing rate	Data type
Past Credit Information I	42%	Numeric
Past Credit Information II	0%	Numeric
Past Credit Information III	18%	Numeric
Family Status	0%	String
Days Past Since Birth	32%	Numeric
Occupation Type	51%	String
Average Area of Apartment	31%	Numeric
Average Area of Common Area	0%	Numeric

TABLE II
EXPERIMENT RESULT

Feature Method	Random Forest	Logit Regression	LightGBM
Original	0.684	0.706	0.721
Polynomial	0.601	0.720	0.733/0.738*
Expert Knowledge	0.677	0.703	0.755/0.778*
FeatureTools	0.681	0.711	0.724
Result with * is recorded after dropping 20 least correlated features.			

Forest model is negatively correlated with the feature number of datasets. This may result from high dimensional sparse features that tend to lead to generating imbalanced trees in Random Forest models.

For the Logistic Regression model, it performs best when trained on the dataset generated through the polynomial approach. This approach applies polynomial calculations to the four features with the highest PCC with the target feature (See in Figure 2). Opposed to the Random Forest model, our result shows that the performance of the Logistic Regression model is better on datasets with more features, such as the polynomial approached generated dataset and the FeatureTools Toolkit generated dataset. Logistic Regression model trained on polynomial approach generated dataset outperform the model trained on the original dataset by 1.4% of AUC score, and model trained with FeatureTools Toolkit generated dataset outperforms the original dataset by 0.5% of AUC score. These two approaches generate features by applying basic polynomial calculations to original features. They can effectively increase the number of features, but at the same time bring problems of multicollinearity. However, the Logistic Regression model relies on a large number of samples and features to improve its performance and this model is not very sensitive to the problem of collinearity, therefore the combination of polynomial generated features and Logistic Regression model is a good choice for credit scoring modeling.

For LightGBM model, it performs best when trained on expert knowledge generated data. The model trained on expert knowledge generated data outperform the second-best LightGBM model trained on polynomial generated data by 2.2% of AUC score, indicating that expert generated features still have an irreplaceable position in credit scoring modeling. The performance of the LightGBM model outperforms the Random Forest model and Logistic Regression model no matter what dataset the model is trained on. The performance of Light GBM model raises when we drop 20 features with least PCC to the target feature, but the performance did not raise much when we drop 10 more least correlated features.

TABLE III
FEATURE NUMBER AFTER FEATURE GENERATION

Original dataset	Polynomial generated dataset	Expert knowledge generated dataset	FeatureTools Toolkit generated dataset
240	274	249	268

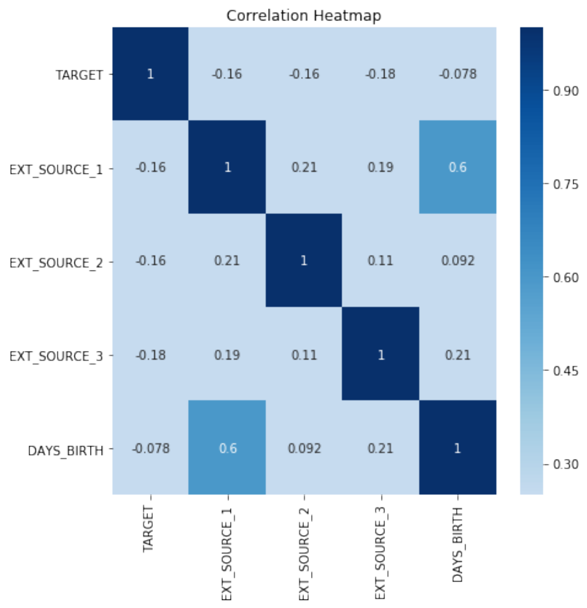

Fig. 2. Heat Map of Features with the highest PCC

VI. CONCLUSION

In conclusion, this paper has compared a variety of approaches of credit scoring modeling, and the experiments have shown that LightGBM model trained on expert knowledge generated features can outperform other models and achieves about 78% in AUC score. For credit risk scoring models, no matter what feature construction method is applied, generally the model performance is LightGBM >Logistic Regression >Random Forest. As for feature construction methods, each method shows its strength when trained by different models. For LightGBM models, expert knowledge approach generated features perform better than other approaches; For Logistic Regression models, polynomial approach generated features works best; For Random Forest model, the original dataset without constructed feature outperform others.

In future work, to improve the performance of proposed models, more machine learning models and techniques will be explored, such as Deep Forest and model stacking. Also, we are planning to experiment proposed models with more credit scoring datasets to verify their robustness.

VII. Acknowledgement

This work is partially supported by the AI University Research Centre (AI-URC) through XJTLU Key Programme Special Fund (KSF-P-02) and KSF-A-17. And it is also partially supported by Suzhou Science and Technology Programme Key Industrial Technology Innovation programme with project code SYG201840. We appreciate their support and guidance.

References

[1] Lyn C. Thomas. A survey of credit and behavioural scoring: forecasting financial risk of lending to consumers. *International Journal of Forecasting*, 16(2):149–172, 2000.

[2] Shweta Arya, Catherine Eckel, and Colin Wichman. Anatomy of the credit score. *Journal of Economic Behavior Organization*, 95(4):175–185, 2013.

[3] Edward I. Altman, Giancarlo Marco, and Franco Varetto. Corporate distress diagnosis: Comparisons using linear discriminant analysis and neural networks (the italian experience) . *Journal of Banking Finance*, 18(3):505–529, 1994.

[4] Hongkyu Jo. Bankruptcy prediction using case-based reasoning, neural networks, and discriminant analysis. *Expert Systems with Applications*, 13(2):97–108, 1997.

[5] DESAI, S V., CROOK, N J., J. R. Overstreet, and A G. A comparison of neural networks and linear scoring models in the credit union environment. *European Journal of Operational Research*, 95(1):24–37, 1996.

[6] Rashmi Malhotra and D. K Malhotra. Evaluating consumer loans using neural networks. *Omega*, 31(2):83–96, 2003.

[7] G Zhang, M Y. Hu, B Eddy Patuwo, and D C. Indro. Artificial neural networks in bankruptcy prediction: General framework and cross-validation analysis. *European Journal of Operational Research*, 116(1):16–32, 1999.

[8] David Feldman and Shulamith Gross. Mortgage default: Classification trees analysis. *Journal of Real Estate Finance Economics*, 30(4):369–396, 2005.

[9] Tian Shyug Lee, Chih Chou Chiu, Yu Chao Chou, and Chi Jie Lu. Mining the customer credit using classification and regression tree and multivariate adaptive regression splines. *Computational Statistics Data Analysis*, 50(4):1113–1130, 2006.

[10] A. Steenackers and M. J. Goovaerts. A credit scoring model for personal loans. *Insurance Mathematics Economics*, 8(1):31–34, 1989.

[11] Sjur Westgaardab. Default probabilities in a corporate bank portfolio: A logistic model approach. *European Journal of Operational Research*, 135(2):338–349, 2001.

[12] Bo Wen Chi and Chiun Chieh Hsu. A hybrid approach to integrate genetic algorithm into dual scoring model in enhancing the performance of credit scoring model. *Expert Systems with Applications*, 39(3):2650–2661, 2012.

[13] Amir Khandani, Adlar J. Kim, and Andrew W. Lo. Consumer credit risk models via machine-learning algorithms. *Social Science Electronic Publishing*.

[14] James Max Kanter and Kalyan Veeramachaneni. Deep feature synthesis: Towards automating data science endeavors. In *2015 IEEE International Conference on Data Science and Advanced Analytics, DSAA 2015, Paris, France, October 19-21, 2015*, pages 1–10. IEEE, 2015.