Hindawi
Discrete Dynamics in Nature and Society
Volume 2021, Article ID 6077540, 10 pages
https://doi.org/10.1155/2021/6077540

Research Article

Automatic Feature Engineering-Based Optimization Method for Car Loan Fraud Detection

Jian Yang, Zixin Tang, Zhenkai Guan, Wenjia Hua, Mingyu Wei, Chunjie Wang, and Chenglong Gu

School of Computer, Nanjing University of Posts and Telecommunications, Nanjing 210023, China

Correspondence should be addressed to Jian Yang; yangj@njupt.edu.cn

Received 9 October 2021; Revised 21 October 2021; Accepted 23 October 2021; Published 7 December 2021

Academic Editor: Gengxin Sun

Fraud detection is one of the core issues of loan risk control, which aims to detect fraudulent loan applications and safeguard the property of both individuals and organizations. Because of its close relevance to the security of financial operations, fraud detection has received widespread attention from industry. In recent years, with the rapid development of artificial intelligence technology, an automatic feature engineering method that can help to generate features has been applied to fraud detection with good results. However, in car loan fraud detection, the existing methods do not satisfy the requirements because of overreliance on behavioral features. To tackle this issue, this paper proposed an optimized deep feature synthesis (DFS) method in the automatic feature engineering scheme to improve the car loan fraud detection. Problems like feature dimension explosion, low interpretability, long training time, and low detection accuracy are solved by compressing abstract and uninterpretable features to limit the depth of DFS algorithm. Experiments are developed based on actual car loan credit database to evaluate the performance of the proposed scheme. Compared with traditional automatic feature engineering methods, the number of features and training time are reduced by 92.5% and 54.3%, respectively, whereas accuracy is improved by 23%. The experiment demonstrates that our scheme effectively improved the existing automatic feature engineering car loan fraud detection methods.

1. Introduction

Car loans, with the characteristics of low threshold, small amount, high liquidity, short cycle, and so forth, have become an important part of online loans. However, the car loan business will certainly face the three following risks: fraud risk, credit risk, and postloan risk. Fraud risk refers to whether the car loan business carried out by the platform has the possibility of attracting fraud gangs to cheat loans. Credit risk refers to whether a single borrower who buys a car has repayment ability. Postloan risk refers to the ability of the platform to dispose of assets after being overdue. Thus, in recent years, due to the continued increase in business volume, optimizing fraud detection, solving a series of problems in credit application fraud, financial intermediary identification, ganging monitoring, or early warning, and building an antifraud cloud platform by means of artificial intelligence methods to improve risk control capability have become a prevailing topic.

In the evolving machine learning methods, most researchers focus on using those state-of-the-art technologies to solve the risk control problem. Neural network and support vector machine [1] showed outstanding performance in specific fraud detection tasks [2]. Yang et al. [3] proposed a "card library reconciliation-preprocessing-neural network detection" workflow to secure the campus card funds from frauds, where the algorithm can detect campus cards with abnormal transactions in the system. Taha and Malebary [4] proposed a new OLightGBM method on the basis of LightGBM [5] algorithm, incorporating the Bayes-based hyperparameter optimization algorithm to achieve credit card fraud detection. The detection accuracy of this method reaches 98.40% on real datasets, which include both fraudulent transactions and legitimate transactions.

This paper studies how to improve the efficiency of fraud detection based on a real car loan dataset. We noticed that traditional fraud detection methods are often limited to batch processing of occurring transactions. During the

computation, the two most central stages in machine learning, feature engineering and training process, are very time-consuming, while different algorithms can have advantages and disadvantages in performance. Thus, the high time cost increases the difficulty for the real-time applications to achieve accurate detection results. Focusing on this problem, Thennakoon et al. [6] proposed a real-time credit card fraud detection method. Similar methods collaborate with software to inform users at the moment when fraudulent transactions occur through the GUI; personally, I prefer using "provides more" time for lending institutions to take relevant measures. However, such method only optimizes the business process of fraud detection applications in engineering. Artificial intelligence fraud detection methods stay unchanged. Based on a car loan dataset, we work on utilizing feature engineering to explore data relationships and improve the accuracy and interpretability of the model at the algorithm level. Currently, similar research directions comprise using multiple transformations to adjust existing features and create new features through the association of different datasets [7], which play an important role in car loan fraud detection. In practical applications, efficiently extracting useful features from large transaction data is extra essential. For instance, Bahnsen et al. [8] used von Mises distribution to analyze the periodic behavior of transaction time on the basis of transaction aggregation strategy. The periodic features were applied to several popular credit card fraud detection models. The results showed that the model cost decreased by 13% in average. Wedge et al. [9] proposed a method based on automatic feature engineering to solve the false positive problems in fraud detection, which used the DFS (deep feature synthesis) algorithm to automatically extract a large number of behavioral features from historical transaction data. 237 features are constructed for each transaction and the method learned the classifier thorough random forest. This method performed well on large-scale datasets and reduced the false alarm rate by 54%. Chen et al. [10] proposed a new neural structure NFS (Neural Feature Search), of which a controller based on recurrent neural network is used to transform each original feature through a series of conversion functions. This controller is trained through reinforcement learning. The method outperforms existing automatic feature engineering methods on public datasets, and it can well reduce the time cost, cutting machine learning development time by a factor of 10, and build better predictive models and generate more meaningful features, while also providing better model performance, preventing unnecessary data leakage, and effectively extracting potentially valuable higher-order transformations, alleviating the feature explosion problem.

According to the researches mentioned above, compared with traditional feature engineering, automated feature engineering can often construct features without limiting the depth, so that it is able to take advantage of feature-based behavioral datasets in fraud detection, which are dominated by features of Boolean types, indicating whether some behavior is done. For example, in financial fraud loan datasets, it relates to whether customers carry out credit rating every year or repay the loan on schedule. The forecasting model is also optimized while saving costs and improving the accuracy simultaneously [11].

In recent years, numerical data has become increasingly popular by virtue of its characteristics of being readily available and easy handling. The financial industry mainly deals with numerical data, and there are many statistics directly related to property like asset, cost, amount of sanction loan, different kinds of ratio, and so on, which totally fit the characteristics of datasets mentioned in this essay. Existing automatic feature engineering methods have problems such as large feature scale, complex model, long training time, and lack of regular measures when processing datasets consist mainly of numerical statistics. Compared with a wide variety of trained models, these numerical datasets with their own characteristics obviously require us to spend more time. A correct understanding of the dataset is essential for subsequent model training and related processing operations. The reason is that the numerically dominated dataset lacks strong explanatory behavioral information, and it has more abstract information and a high degree of quantification. Compared with the behavioral dataset, numerical datasets are more difficult to predict. Meanwhile, numerical datasets in financial sector also bring about difficulty in processing missing values and exception values and finding deep connection between relevant features, which has something to do with the final result. It is also an important research topic for future loan fraud detection products with high potential engineering value.

A car loan dataset characterized mainly in numerical data is studied in this paper, and the existing automatic feature engineering methods are optimized by limiting the depth of feature generation to solve the above problems. The advantages can be summarized as follows:

(1) Enhanced feature interpretability. The number of features is reduced by 92.5%, which is from originally 1,520 features to 114 features, so that the features can be faster processed. Hence, the model can be trained based on cognition, avoiding unexplainable abnormalities when the performance of the model decreases due to complex features.

(2) Reduced total time cost. Although the existing automatic feature engineering works by shortening the time to process features, the substantial time cost increase in training stage still has large weight in the overall solution. The performance of the proposed method in this paper shortens the time of feature engineering and model training by 54.3% compared with traditional automatic feature engineering methods.

Experiment results demonstrate that the proposed automatic feature engineering method improves detection accuracy by 23% in car loan fraud detection. In addition, optimized performance can be seen in base model selection, automatic feature generation, optimization, and time cost control.

2. Materials and Methods

2.1. Deep Feature Synthesis. From rule-based to feature-based, the dimensionality of machine learning continues to increase in problem solving, while the reliance on rules and experience continues to decrease. For a given dataset, feature

engineering is the last step before the data enters the model. Whether it is handled properly will affect the performance of the model to a large extent. Hence, there is no denying that feature engineering is a key part of the data processing flow based on machine learning. From an engineering perspective, Amazon's chief scientist Li Mu believes that "the time spent on data for machine learning projects should account for more than 80% of the total time [12].

Generally, there is a disproportion between positive and negative samples in fraud detection datasets; and fraud samples often only occupy a small part of the sample space. The value of feature engineering in fraud detection problems lies in amplifying the features of positive samples and detecting them more accurately. In pursuit of good engineering effects, in-depth domain knowledge and manual processing, if necessary, are required in feature engineering, which greatly affects the efficiency of smart fraud detection products. Automated feature engineering simplifies the cumbersome work, laying a foundation for optimizing the construction and deployment of machine learning models, which frees data scientists from complex feature engineering work. The involvement of automatic feature engineering, especially the invention of the open-source tools like Featuretools [13], improved the overall efficiency of smart fraud detection data products and has attracted more and more attention.

2.2. Application of Automatic Feature Engineering Method in Car Loan Fraud Detection. In this paper, automatic feature engineering method is used in car loan fraud detection.

Around this specific application scenario, the method has the following advantages:

(1) Automatic feature engineering provides a quantitative method of features, eliminating some meaningless feature engineering operations.

(2) Automatic feature engineering provides a method to manipulate features in batches with higher efficiency, reducing the time cost significantly compared with selecting features manually.

(3) Generally speaking, domain knowledge is required in feature engineering. Automatic feature engineering reduces the dependence on domain knowledge and facilitates mutual collaboration.

Compared with fraud detection methods based on knowledge graph [14], the data utilization rate is higher in automatic feature engineering. In both the precision, which indicates the capacity of the model to avoid the inclusion of samples from any other classes in the analyzed class, and the recall, which shows the capacity of the model to include all the samples that, in fact, are inside a class, automatic feature engineering works better than traditional feature extraction techniques. The average accuracy reaches 94.96% [15], while valuable information can be extracted from fewer features in addition.

The existing automatic feature engineering still has shortcomings, manifested in the following aspects:

(1) The number of features extracted through automatic feature engineering is too large. As a result, the time saved in feature engineering is offset by the high cost of model training and the cost of model tuning time soars. The practical application value of automatic feature engineering is limited by the increased total cost.

(2) Feature dimensional explosion [16] leads to a decline in the interpretability of new features. One of the main contributions of this paper can be concluded to solve the problems of feature superposition, depth loss, and interpretability decline by limiting the depth. The original features are manually filtered so that the newly generated features have a certain directionality, which ensures the accuracy of the new features while reducing the depth.

(3) If the model stays unadjusted, too many automatically generated new features can easily lead to adverse effects on dirty data; the noise in the dataset will also affect the performance of the model.

Automatic feature engineering lacks regular measures when adding features. Too many constructions of the same feature can easily result in excessive rewards for a few features, which makes it too weighted in the model, leading to overfitting at the end. In fraud detection, fraudsters will disguise themselves. When the confidence of few samples is too high, it is difficult to identify fraudulent behaviors.

The car loan dataset studied in this paper is a dataset whose characteristic types are mainly numerical. As mentioned above, existing automatic feature engineering methods have a large number of features, high model complexity, high training time cost, and lack of regularity. These methods cannot achieve the expected effect on the car loan dataset. This paper attempts to limit the depth of feature generation through strategies and optimize the method of automated feature engineering.

2.3. Analysis of Car Loan Fraud Detection Data

2.3.1. Preprocessing. The dataset used in this article is provided by a domestic financial institution and contains 52 fields. This paper selects 150,000 pieces of data as the research sample. The dataset has the following characteristics:

(1) Disproportionate data samples: 26,545 out of 150,000 records are predicted to be fraudulent, which are referred to as positive samples. That is, 123,455 records are labeled as negative samples, accounting for 82.4% of the total sample space. Actually, data imbalance is a property of this dataset itself and a natural law of car loan samples. There are about 500,000 original pieces of data, and we screened 150,000 of them and retained the imbalance property appropriately.

(2) Some highly correlated features: a heat map generated with relation coefficients of all features (Figure 1) was used to detect the relationship between each original feature and the degree of

Figure 1: Relation coefficients heat map.

connection between the original feature and the predicted value. The selected features form the x-axis and y-axis, and the correlation coefficients of each pair of features are given by the following formula:

$$r = \frac{\sum (X - \overline{X})(Y - \overline{Y})}{\sqrt{\sum (X - \overline{X})^2 \sum (Y - \overline{Y})^2}} \sqrt{b^2 - 4ac}. \quad (1)$$

(3) Numerical features dominated: several typical characteristics are selected for analysis (Table 1).

The above features contain some characteristics:

(1) The features in the data are basically numeric data.

(2) Among the numerical types, different types are also included, such as some data indicating economic quantities, such as asset cost; some data indicating ratios, such as the following features with ratio; and some data showing customer attributes, such as Credit_score.

These features cover almost all the characteristics of the whole data and are very useful to provide ideas for our subsequent processing.

2.3.2. Classification. Sufficient training samples are conducive to the improvement of model performance. We selected 80% of the 150,000 datasets as the training set and

Table 1: Typical characteristics.

Characteristic name	Characteristic description
Customer_id	Customer identifier
Main_account_outstanding_loan	The outstanding loan balance of the master account
Disbursed_amount	Amount of disbursed loan
Asset_cost	Cost of assets
Credit_score	Credit score
Loan_to_asset_ratio	Loan-to-asset ratio
Outstanding_disburse_ratio	Total loans disbursed/total loans outstanding (ratio of the two)
Active_to_inactive_ratio	Number of valid loans/number of invalid loans (ratio of the two)

20% as the test set. While avoiding underfitting, try to choose a large test set. Experiments show that the training of the model is successful.

2.4. Deep Feature Synthesis Algorithm. Deep feature synthesis is an algorithm that automatically generates features for relational datasets, proposed by Kalyan Veeramachaneni and Max Kanter of the MIT Computer Science and Artificial Intelligence Laboratory [17]. The algorithm can automatically construct a predictive model for complex datasets. As an algorithm that automatically generates features for relational datasets, DFS follows the relationship between the data and the basic fields and sequentially applies mathematical functions along the path to create the final features. Through sequential stacking calculations, experiments show that each new feature can be defined as a specific depth.

2.4.1. Entity Feature. Entity feature (efeat) calculates the value of each entry to derive characteristics. Based on element-wise multiplication, these features can be applied to the calculation function arrays x and j. Examples include converting existing features in the entity table into another type of value function, such as converting the categorical string data type to a predetermined unique value or rounding the value. Other examples include converting timestamps into 4 different characteristics: working days (1–7), days in a month (1–30/31), number of months of the year (1–12), and hours (1–24). The data used in this paper converts discrete features such as manufacturer number and service personnel number into one-hot coding features or converts the mobile phone number fill-in feature into Boolean type features.

This function is applied to the entire value set of the j-th feature $x_{:,j}$ and $x_{I,j}$:

$$x_{i,j'} = e\,\mathrm{feat}\left(x_{:,j,i}\right). \tag{2}$$

2.4.2. Relation Feature. Relation features are obtained through joint analysis of two related entities and E^k, including forward and backward relations:

Forward: it is defined between an instance m of entity E^l and another instance i of entity E^k. This is considered as a forward relationship because i has an explicit dependency in m.

Backward: backward relation refers to the relationship from an instance i in E^k to all instances $m = \{1 \ldots M\}$ in E^l which have forward relationship to k.

Direct features (dfeat) are applied over the forward relationships. In these, features in a related entity $i \in E^k$ are directly transferred as features for $i \in E^k$.

Relational features (rfeat) are applied over the backward relationships. They are derived for an instance i of entity E^k by applying a mathematical function to $x^l_{:,j|e^k=i}$, which is a collection of values for feature j in related entity E^l, assembled by extracting all the values for feature j in entity E^l where the identifier of E^k is $e^k = i$. This transformation is expressed as

$$x^k_{i,j'} = r\,\mathrm{feat}\left(x^l_{:,j|e^k=i}\right). \tag{3}$$

Some examples of rfeat functions are min, max, and count. rfeat functions could also be applied to the probability density function over $x^l_{:,j|e^k=i}$.

2.5. Algorithm and Feature Analysis

2.5.1. Feature Synthesis Abstractions. The deep feature synthesis process based on the algorithm of DFS is shown in Figure 2.

2.5.2. Deep Feature Synthesis Algorithm. Deep feature synthesis is based on the data table to add features to the target entity. The detailed algorithm is as follows:

(1) Aggregate the entity table. For each entity feature in the table, expand the dfeat and rfeat.

(2) For the derived dfeat, perform the second round of feature synthesis through forward relation. For the derived rfeat, perform feature synthesis several times after iterating.

(3) Get all the features of deep feature synthesis.

The pseudocode of the algorithm is given in Algorithm 1.

2.5.3. Growth of Number of Features. The feature space that can be enumerated by deep feature synthesis grows very quickly. In this paper, we analyze the number of features, z, and the algorithm will synthesize for a given entity. Due to the recursive nature of feature synthesis, the number of features created for an entity depends on the number created for related entities. Thus, we use z_i to represent the number

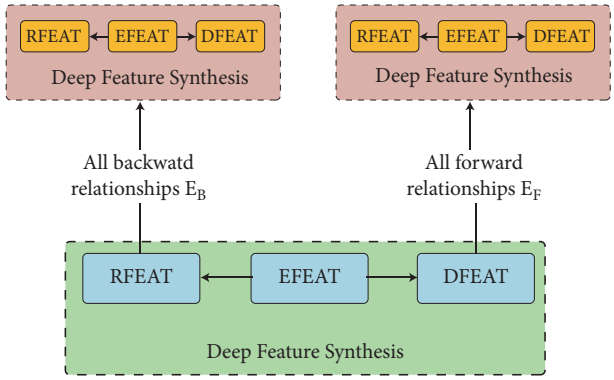

Figure 2: Deep feature synthesis process.

of features we create for an entity if we recurse i times. Assume that all entities in our dataset start with $O(j)$ features, $O(n)$ forward relationships, and $O(m)$ backward relationships.

First, synthesize rfeat features for entity $O(m)$. Suppose that there are $O(r)$ rfeat functions; then we synthesize $O(r \cdot z_{i-1})$ features for each of the m entities in a backward relationship for a total of $O(r \cdot z_{i-1} \cdot m)$ additional features. Second, generate one dfeat feature for every feature in entities in a forward relationship, which means $O(z_{i-1} \cdot n)$ features are added. Finally, make efeat features with j original features and $O(z_{i-1} \cdot (r \cdot m + n))$ new constructed features. Assume that there are $O(e)$ efeat features in total; then the final number of efeat features is $O(e \cdot j + e(z_{i-1} \cdot (r \cdot m + n)))$.

Combining all rfeat, dfeat, and efeat features, we see that

$$z_i = O\left(z_{i-1} \cdot (r \cdot m + n)(e + 1) + e \cdot j\right). \tag{4}$$

If $i = 0$, only efeat features can be calculated, so

$$z_0 = O(e \cdot j). \tag{5}$$

Let $p = (r \cdot m + n)(e + 1)$ and $q = e \cdot j$; by substitution, we get

$$z_i = O\left(z_{i-1} \cdot p + q\right). \tag{6}$$

Replacing z_{i-1} by $z_{i-2} \cdot p + q$, we get

$$z_i = O\left(z_{i-2} \cdot p^2 + q \cdot p + q\right). \tag{7}$$

Continuing the iteration till z_0, we get

$$z_i = O\left(z_0 \cdot p^i + q \cdot p^{i-1} + q \cdot p^{i-2} \cdots + q\right). \tag{8}$$

Replacing in the above equation, we get

$$\begin{aligned} z_i &= O\left(q \cdot p^i + q \cdot p^{i-1} + q \cdot p^{i-2} \ldots + q\right), \\ z_i &= O\left(q \cdot \left(\sum_{u=0}^{i} p^u\right)\right). \end{aligned} \tag{9}$$

Therefore, the closed form for z_i is

$$z_i = \left(O(e \cdot j) \sum_{u=0}^{i} (r \cdot m + n)^u \cdot (e + 1)^u\right). \tag{10}$$

2.6. Feature Selection

2.6.1. Feature Primitives. Primitive features are the most basic feature granularity. In this paper, new features can be constructed based on primitive features. There are two sources of primitive features shown as follows:

(1) Aggregation: the production process involves variables produced by the integration of multiple features, such as the maximum value, minimum value, and mean value in the superimposed feature.

(2) Transformation: generate feature variables by operating only in a single feature.

2.6.2. Choose Features. In this paper, the DFS algorithm is used to generate features. Two generation methods, namely, aggregation and transformation based on metafeatures, are implemented. We adjusted for the parameters of the algorithm. If no constraint is placed on the depth of the algorithm, the relation features will operate with other features many times, making the algorithm go deeper and deeper. When we limit the depth of the algorithm to one or two layers, the frequency of the operation decreases, so the depth of the algorithm can also decrease. In addition, we also reduced the number of features involved in feature engineering to ensure that invalid features are not overincluded in feature engineering.

(1) Based on the aggregation method, the generated features in the fields provided by the dataset are shown in Table 2.

(2) Based on the transformation method, the generated features in the fields provided by the dataset are shown in Table 3.

(3) Different feature generation algorithms come with different time and space complexity, so the choice of method is important. If choosing multiply_numeric, the final generated features are shown in Table 4.

3. Results and Discussion

3.1. Configuration and Indicators

3.1.1. Experiment Environment. The server configuration of the experiment is as follows: Intel(R) Core(TM) i7-10875H CPU @ 2.30 GHz and GeForce RTX 2070 Super GPU. The programming languages are *Python* 3.8 and scikit-learn 0.24.2. Jupyter notebook works as the compiler.

3.1.2. Evaluation Index. Three model evaluation indicators are considered in this paper: accuracy, receiver operating characteristic curve (ROC), and area under ROC curve (AUC).

Accuracy refers to the accuracy of the model. ROC curve and AUC coefficient are mainly used to test the ability of the model to correctly rank customers. The ROC curve describes the cumulative ratio of negative customers under a certain proportion of positive customers. The more robust the model is to distinguish, the closer the ROC curve is to the

(1) **function** $\text{MAKE}_{\text{FEATURES}(E^I, E^{1:M}, E_V)}$
(2) $E_V = E_V \cup E^i$
(3) $E_B = \text{BACKWORD}(E^I, E^{1\ldots M})$
(4) $E_F = \text{FORWORD}(E^I, E^{1\ldots M})$
(5) **for** $E^j \in E_B$ **do**
(6) $\text{MAKE}_{\text{FEATURES}(E^j, E^{1\ldots M}, E_V)}$
(7) $F^j = F^j \cup \text{RFEAT}(E^I, E^j)$
(8) **for** $E^j \in E_F$ **do**
(9) **if** $E^j \in E_V$ **then**
(10) CONTINUE
(11) $\text{MAKE}_{\text{FEATURES}(E^j, E^{1\ldots M}, E_V)}$
(12) $F^i = F^i \cup \text{DFEAT}(E^I, E^j)$
(13) $F^i = F^i \cup \text{EFEAT}(E^i)$

Algorithm 1: Generate features for the target entity.

Table 2: Features generated through aggregation.

Feature name	Explanation
min	Minimum value of the feature
mean	Mean value of the feature
std	Variance of correlation value
count	Number of recorded values

Table 3: Features generated through transformation.

Feature name	Explanation
cum_sum	Calculate the cumulative maximum
multiply_numeric	Multiply the elements of two lists
add_numeric_scalar	Add a scalar to each value in the list

upper left corner. The area under the ROC curve is represented through AUC coefficient. The higher the AUC coefficient, the stronger the risk discrimination ability of the model. ROC curve and AUC coefficient are commonly used indicators to measure the pros and cons of risk control models, which are very suitable for evaluating the effect of fraud detection.

The results and discussion may be presented separately, or in one combined section, and may optionally be divided into headed subsections.

3.2. Experiments and Experimental Results

3.2.1. Comparison of the Base Models. Several mainstream models in machine learning are selected for pretraining. Through K-fold cross-validation [18], the average value of K prediction accuracy of the model is used to evaluate the predictive ability of the model.

The pretraining results are shown in Figure 3.

Among the base models above, RandomForestClassifier [19], GradientBoostingClassifier [20], ExtraTreeClassifer, and LightGBM achieved roughly the same results. To compare the training times of these four models, results are shown in Table 5.

It is evident that LightGBM takes the least time in training, improving the efficiency by 74.5% compared with ExtraTreeClassifer. The dataset in this paper contains

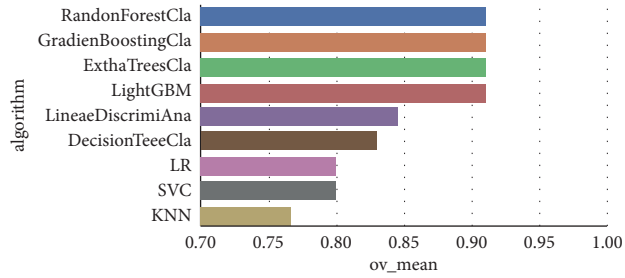

Figure 3: Base model performance comparison.

150,000 pieces of records, while the amount of data credit institutions need to process in actual loan fraud is far more than that. It is reasonable to choose the LightGBM algorithm with the shortest training time.

3.2.2. Experiment of Automatic Feature Engineering. In this experiment, the results of the proposed scheme are compared with a benchmark group and a group with existing method. The settings of each group are shown in Table 6.

After experiments, the comparison of the experimental groups is shown in Table 7.

The ROC curves of the three experiment types are shown in Figures 4–6.

The abscissa of ROC curve is false positive rate (FPR). The ordinate of ROC curve is true positive rate (TPR).

FPR indicates how many of all negative cases are predicted to be positive, and TPR indicates how many real positive examples are predicted.

Here the calculation method of ROC is given:

(1) Sort from large to small according to score the probability that each test sample belongs to a positive sample.

Table 4: Generated features.

Feature name	Classification	Explanation
modulo_numeric_scalar	Transformation	Return the modulus of each element in the list through a scalar
divide_by_feature	Transformation	Divide the scalar by each value in the list.

Table 5: Training time comparison.

Model	Time cost (seconds)
RandomForestClassifier	54.3
ExtraTreeClassifer	37.27
GradientBoostingClassifier	75.4
LightGBM	9.50

Table 6: Settings of experimental group.

Group	Description
Benchmark group	Use original features
Automatic feature group A	Use automatically constructed features that do not limit depth
Automatic feature group B	Use automatically constructed features with limited depth

Table 7: Experimental performance comparison.

Group	Number of features	Construction time	Training time	Model training accuracy
Benchmark group	39	—	1 min and 42 sec	0.64
Automatic feature group A	1520	5 min	33 min and 28 sec	0.66
Automatic feature group B	114	15 min	2 min and 33 sec	0.86

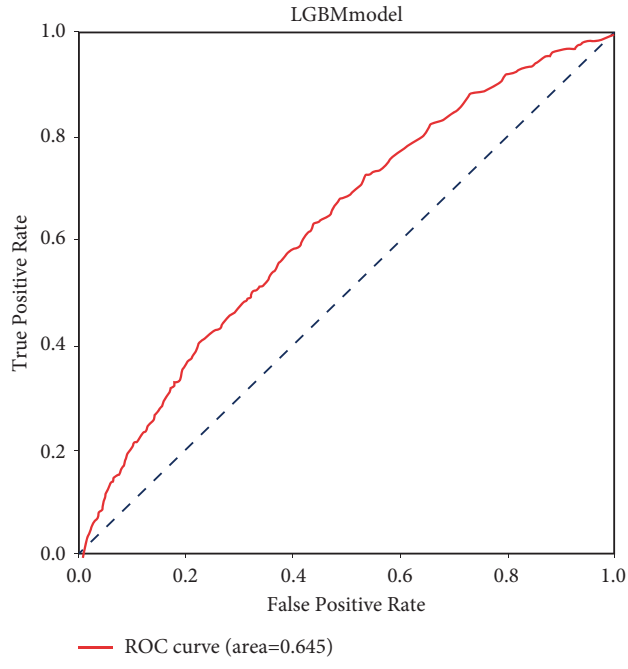

Figure 4: ROC curve of the benchmark group.

(2) From high to low, take score as the threshold. When the probability of the test sample belonging to the positive sample is greater than or equal to this threshold, we consider it as a positive sample; otherwise, it is considered as a negative sample.

(3) Each time we select different scores as the threshold, we can get a set of FPR and TPR, that is, a point on the curve. A total of 20 groups of FPR and TPR values are obtained. We can obtain a complete ROC curve by connecting these (FPR and TPR) pairs.

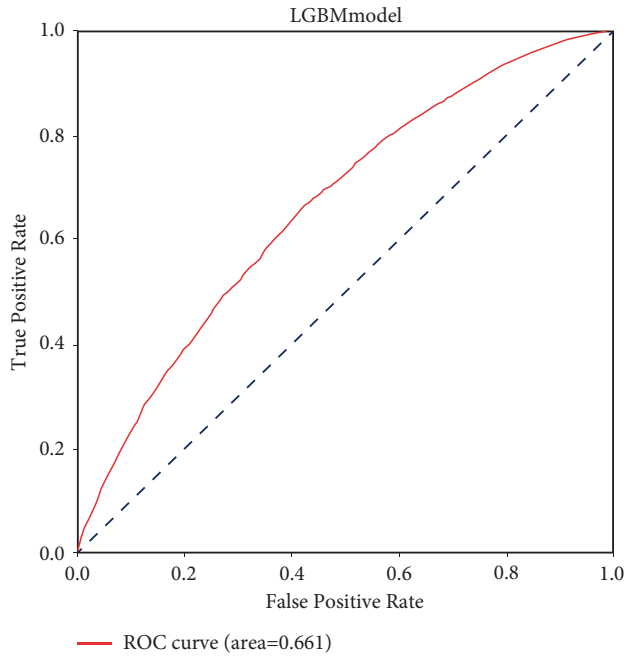

Figure 5: ROC curve of the automatic feature group A.

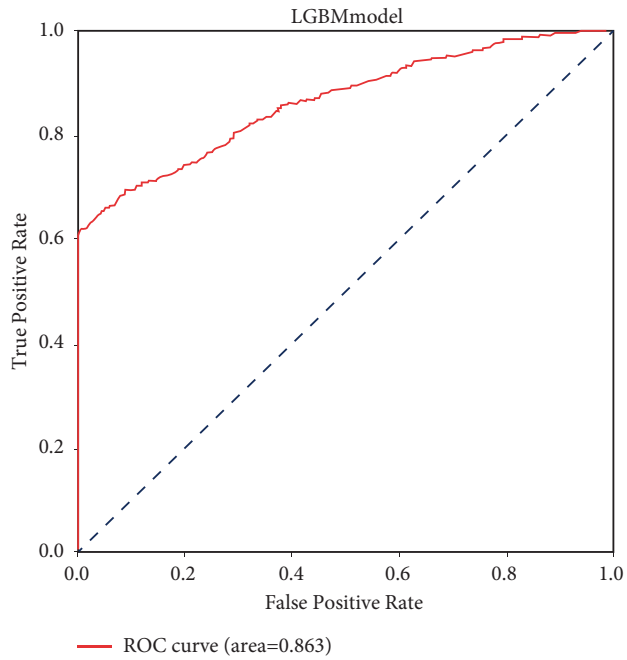

Figure 6: ROC curve of the automatic feature group B.

4. Conclusions

After analyzing the results, we can draw the following conclusions:

(1) By comparing the ROC curve and AUC coefficient of all the groups, it is shown that the automatic feature group B performs better. 23% of the corresponding AUC value of group B is optimized compared with group A, and 25.5% optimized compared with the benchmark group.

(2) Compared with the automatic feature group A, the automatic feature group B shortens the total construction time and training time by 54.3%. The number of generated features is reduced by 92.5%. The readability and interpretation ability are optimized.

(3) The goal of automatic feature engineering is to shorten the time spent on feature engineering and thereby allocate more time to model tuning and other steps. In the automatic feature group A, the long training time actually adds time cost, which goes against the goal. The automatic feature group B shortens the time spent on feature engineering, while the training cost does not increase significantly, which further approaches the actual goal of automatic feature engineering.

5. Discussion

Automatic feature engineering has a wide range of applications in the field of data science. In this paper, deep feature synthesis algorithms are used to improve the effect of car loan fraud detection. The depth of DFS algorithm is limited by compressing abstract and lacking interpretative features. The following problems are solved: feature dimension explosion, low interpretability, low interpretability, complex features, and other issues. Compared with traditional automatic feature engineering methods, the method proposed in this paper reduces the number of features by 92.5%. The training time is shortened by 54.3%, while the detection accuracy is increased by 23%. The existing automatic feature engineering car loan fraud detection method is effectively optimized. It is worth noting that when the numerical features in the dataset are not dominant or the features are sparse, deep automatic feature synthesis is still valid. Our future work is to study how to optimize the capacity of the model and improve the operating efficiency under this condition.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] W. S. Noble, "What is a support vector machine?" *Nature Biotechnology*, vol. 24, no. 12, pp. 1565–1567, 2006.

[2] J. West and M. Bhattacharya, "Intelligent financial fraud detection: a comprehensive review," *Computers & Security*, vol. 57, pp. 47–66, 2016.

[3] Y. Yang, X. Zhou, and Z. R. Zhou, "Fraud detection model of campus card," *Computer Engineering*, vol. 37, no. 12, pp. 113–115, 2011.

[4] A. A. Taha and S. J. Malebary, "An intelligent approach to credit card fraud detection using an optimized light gradient

boosting machine," *IEEE Access*, vol. 8, pp. 25579–25587, 2020.

[5] G. Ke, Q. Meng, T. Finley, T. F. Wang, and W. Chen, "LightGBM: a highly efficient gradient boosting decision tree," in *Proceedings of the 31st Conference on Neural Information Processing Systems (NIPS)*, Long Beach, CA, USA, December 2017.

[6] A. Thennakoon, C. Bhagyani, S. Premadasa, and S. Mihiranga, "Real-time credit card fraud detection using machine learning," in *Proceedings of the 9th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*, Noida, India, January 2019.

[7] T. Verdonck, B. Baesens, and M. Óskarsdóttir, "Special issue on feature engineering editorial," *Machine Learning*, 2021.

[8] A. Correa Bahnsen, D. Aouada, A. Stojanovic, and B. Ottersten, "Feature engineering strategies for credit card fraud detection," *Expert Systems with Applications*, vol. 51, pp. 134–142, 2016.

[9] R. Wedge, J. M. Kanter, and K. Veeramachaneni, "Solving the false positives problem in fraud prediction using automated feature engineering," *Lecture Notes in Computer Science*, vol. 11053, 2019.

[10] X. Chen, Q. Lin, C. Luo, and X. Li, "Neural feature Search: a neural architecture for automated feature engineering," in *Proceedings of the 2019 IEEE International Conference on Data Mining (ICDM)*, pp. 71–80, Beijing, China, November 2019.

[11] C. Wang and C. Q. Wang, "An automated feature engineering method for online payment fraud detection," *Chinese Journal of Computers*, vol. 43, no. 10, pp. 1983–2001, 2020.

[12] M. Li, *Dive into Deep Learning*, 2021.

[13] GauravSheni, *MachineFL*, 2021.

[14] Q. Wang, Z. Mao, B. Wang, and L. Guo, "Knowledge graph embedding: a survey of approaches and applications," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 12, pp. 2724–2743, 2017.

[15] D. Cabrera, F. Sancho, C. Li et al., "Automatic feature extraction of time-series applied to fault severity assessment of helical gearbox in stationary and non-stationary speed operation," *Applied Soft Computing*, vol. 58, pp. 53–64, 2017.

[16] P. Indyk and R. Motwani, "Approximate nearest neighbors: towards removing the curse of dimensionality," in *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, Dallas, TX, USA, May 1998.

[17] J. M. Kanter and K. Veeramachaneni, "Deep feature synthesis: towards automating data science endeavors," in *Proceedings of the IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, pp. 1–10, Paris, France, October 2015.

[18] J. D. Rodr´ıguez, A. P´erez, and J. A. Lozano, "Sensitivity analysis of k-fold cross validation in prediction error estimation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 3, pp. 569–575, 2010.

[19] A. Liaw and M. Wiener, "Classification and regression by randomForest," *R News*, vol. 2, no. 3, pp. 18–22, 2002.

[20] C. Bentéjac, A. Csrg, and G. Martínez-Muoz, "A comparative analysis of gradient boosting algorithms," *Artificial Intelligence Review*, vol. 54, no. 8, pp. 1937–1967, 2021.