symmetry

MDPI

Article

AutoEncoder and LightGBM for Credit Card Fraud Detection Problems

Haichao Du [1,2,*], Li Lv [2], An Guo [3,4] and Hongliang Wang [2]

[1] University of Chinese Academy of Sciences, Beijing 100049, China
[2] Shenyang Institute of Computing Technology, Chinese Academy of Sciences, Shenyang 110168, China; lli@sict.ac.cn (L.L.); wanghl@sict.ac.cn (H.W.)
[3] School of Computer & Information Engineering, Anyang Normal University, Anyang 455000, China; guoan@aynu.edu.cn
[4] Key Laboratory of Oracle Bone Inscriptions Information Processing, Ministry of Education of China, Anyang 455000, China
* Correspondence: duhaichao20@mails.ucas.ac.cn

Abstract: This paper proposes a method called autoencoder with probabilistic LightGBM (AED-LGB) for detecting credit card frauds. This deep learning-based AED-LGB algorithm first extracts low-dimensional feature data from high-dimensional bank credit card feature data using the characteristics of an autoencoder which has a symmetrical network structure, enhancing the ability of feature representation learning. The credit card fraud dataset comes from a real dataset anonymized by a bank and is highly imbalanced, with normal data far greater than fraud data. For this situation, the smote algorithm is used to resample the data before putting the extracted feature data into LightGBM, making the amount of fraud data and non-fraud data equal. After comparing the resampled and non-resampled data, it was found that the performance of the AED-LGB algorithm was not improved after resampling, and it was concluded that the AED-LGB algorithm is more suitable for imbalanced data. Finally, the AED-LGB algorithm is comparable with other commonly used machine learning algorithms, such as KNN and LightGBM, and it has an overall improvement of 2% in terms of the ACC index compared to LightGBM and KNN. When the threshold is set to 0.2, the MCC index of AED-LGB is 4% higher than that of the second-highest LightGBM algorithm and 30% higher than that of KNN. It shows that the AED-LGB algorithm has higher performance in accuracy, true positive rate, true negative rate, and Matthew's correlation coefficient.

Keywords: credit card fraud; AED-LGB algorithm; autoencoder; symmetrical network structure

Citation: Du, H.; Lv, L.; Guo, A.; Wang, H. AutoEncoder and LightGBM for Credit Card Fraud Detection Problems. *Symmetry* **2023**, *15*, 870. https://doi.org/10.3390/sym15040870

Academic Editors: Jeng-Shyang Pan and Sergei D. Odintsov

Received: 7 February 2023
Revised: 6 March 2023
Accepted: 31 March 2023
Published: 6 April 2023

1. Introduction

The development of the internet, information technology, and wireless devices has dramatically changed people's lives. Traditional banks are gradually transitioning from bank counters and branch services to online electronic services. Electronic banking, with its convenience and value-added services, has become the main driving force for banks to maintain vitality and competitiveness. Credit cards, born with the development of electronic banking, are widely used all over the world; in the USA alone, there are about 100 billion credit cards currently in use [1]. However, there are various fraudulent methods surrounding electronic channels that cause user information disclosure and fund theft, causing headaches for banks. Traditional fraud detection methods are often based on business experience, expert experience, and the creation of blacklists, but they suffer from single-dimensional coverage problems [2]. The traditional anti-fraud methods are single-dimensional, making it difficult to form a multi-dimensional user portrait and to analyze the customer's behavior preferences, debt repayment ability, payment ability, and fraud tendencies. There is also a high cost of human operation, low efficiency, and high application costs. With the development of artificial intelligence and big data technology, banks are

shifting away from traditional manual detection methods to algorithmic fraud detection. As such, it is particularly important to design an algorithm for fraud anomaly detection.

The development of deep learning allows scientists to use deep learning methods to extract complex nonlinear features [3], thereby promoting the development of machine vision and speech recognition. Inspired by this, this paper proposes using an autoencoder in deep learning to model and detect fraudulent activities in credit card transactions [4]. Unlike other bank detection algorithms, such as the traditional logical regression algorithm, the AED-LGB algorithm combines machine learning and deep learning to achieve more accurate and efficient predictions of fraudulent data, even in imbalanced data scenarios. The overall structure of the AED-LGB algorithm consists of an autoencoder and an LGBM; firstly, the autoencoder is used to train the data and extract feature data, then the feature data is fed into the LGBM algorithm for classification prediction to determine if the credit card transaction data is normal or abnormal. Unlike other methods that adopt a binary (0/1) classification for the LGBM, the AED-LGB adopts LGBM with probabilistic classification, which further improves its performance. Detailed information will be described in the following chapters.

In this article, we aim to validate our algorithm by using a dataset of credit card transactions from a European bank. The dataset includes 284,807 transactions over two days, with 492 of them being fraudulent (accounting for 0.172% of all transactions). The dataset only has numerical input variables, which are the result of a PCA transformation. Due to confidentiality issues, the original features and background information about the data cannot be provided. The features V1, V2, ... V28 are the principal components obtained from PCA, while the only features that have not been transformed by PCA are 'Time' and 'Amount.' 'Time' represents the number of seconds between each transaction and the first transaction in the dataset, and 'Amount' is the transaction amount, which can be used for example-dependent cost-sensitive learning. The 'Class' feature represents the response variable, with a value of 1 indicating fraud and 0 otherwise.

The article has three main contributions. Firstly, the AED-LGB algorithm is introduced, using the autoencoder to extract low-dimensional features from the high-dimensional credit card feature data, improving feature representation learning ability. Secondly, the extracted features are then fed into LightGBM, and due to the unbalanced data, the SMOTE algorithm is used to resample the data so that the amount of fraudulent and non-fraudulent data are the same. The results of the comparison between the resampled and non-resampled data show that the AED-LGB algorithm is better suited for imbalanced data. Lastly, the AED-LGB algorithm is compared with other common fraud detection algorithms, and the results show that AED-LGB performs excellently in terms of accuracy, true positive rate, true negative rate, and Matthew's correlation coefficient. The overall contributions of my work are as follows:

- The proposed AED-LGB model, which combines autoencoder and LightGBM, resolves the issues concerning detecting credit card fraud.
- The proposed AED-LGB uses the LightGBM model with probabilistic classifications to classify data. Fine-tuning the probability threshold θ value can provide AED-LGB with a better classification result.
- The proposed model was evaluated using MCC, TPR, TNR, and ACC values. The experimental results indicate superior performance to that of existing models, such as KNN, LightGBM, and random forest.

The rest of the article is organized as follows. Section 2 introduces the technical research status of credit card fraud detection both domestically and abroad. Section 3 describes the overall network structure and algorithms used in our AED-LGB algorithm, as well as the algorithm flow. Section 4 contains the experiments. Section 5 concludes the research.

2. Related Work

The rapid development of electronic payments has made credit card payments the preferred choice for young people when shopping. Its advantage of pre-payment and installment-based consumption solves the problem of young people not being able to purchase the products they like due to **a** temporary shortage of funds. However, some unscrupulous users attempt to misuse the normal behavior of credit card payments by maliciously over-drafting other people's cards. These people are often reticent to repay their balance, which poses a great financial risk to banks and other financial institutions. To address this issue, many researchers in the industry have tried using machine learning and deep learning algorithms to detect credit card fraud.

There are many related algorithms in the field of anomaly detection [5,6], and there are relevant research reports that highlight the challenges of credit card fraud detection, including the problem of data imbalance, where the number of positive samples is much larger than the number of negative samples, and the model is prone to overfitting during training [7,8]. The last issue is the neglect of the specific relationship between the continuous features of time-series data, which machine learning algorithms often focus on individual features instead of whole sequences of transactions. To address these issues, researchers in the industry have used corresponding techniques [9]. For example, to solve the imbalance problem, some authors have used data resampling algorithms [10,11] to achieve ideal results [12]. To address the neglect of continuous features, some researchers have used deep learning technology and verified its effectiveness [13,14]. Currently, the algorithms for credit card fraud detection in banks are mainly machine learning algorithms [15,16]. Machine learning algorithms are divided into supervised and unsupervised learning. Supervised learning includes random forest, logistic regression [17,18], LightGBM, etc.; the classic non-clustering algorithms of supervised learning are KNN [19]. These methods have also achieved good results in some aspects. There are also some classic algorithms for unsupervised learning, such as autoencoders [20] and clustering [21]. The k-nearest neighbor (KNN) method was initially proposed by Cover and Hart in 1968 and is one of the simplest machine learning algorithms. It belongs to the classification algorithm in supervised learning. N. Malini et al. [19] conducted an analysis of credit card fraud identification techniques based on KNN and outlier detection. They perform oversampling and extract data and then use the KNN method to determine the abnormality of the target instance. The KNN method is suitable for detecting frauds under the condition of limited memory, and the anomaly detection mechanism helps to detect credit card frauds with less memory and computational requirements. Especially, anomaly detection runs faster and better on large datasets. Finally, by comparing with other known anomaly detection methods, the experimental results show that the KNN method is accurate and effective. Logistic regression is a machine learning technique for solving binary classification (0 or 1) problems and is used to estimate the probability of something. At present, many banks are also using logistic regression for the scorecard model, credit card fraud detection, and other applications, and have achieved good results.

Random forest is a classifier that uses multiple trees to train and predict samples. M.VamsiKrishna [22] proposed the use of random forest and logistic regression algorithms for comparison experiments and found that random forest has achieved higher accuracy than logistic regression in credit card detection comparison. However, when the data is imbalanced, the models trained by KNN, LightGBM, and random forest algorithms have relatively low MCC metrics on the test set.

Some researchers have proposed semi-supervised methods [23] based on autoencoders for anomaly detection [24,25]. These methods involve training deep autoencoders on data samples without anomalies and detecting anomalous events based on the reconstruction errors of anomalous and normal samples. Autoencoders can be used to learn better representations of samples, thus improving the effectiveness of classification [26]. In light of the above, this paper proposes a combining autoencoder and LightGBM algorithm,

and through the training of this model, our algorithm shows better performance on the above datasets.

3. The Proposed Method

As just mentioned, the proposed AED-LGB method uses the autoencoder to extract data features and applies the LightGBM with probabilistic classification to classify credit card transactions as normal or fraudulent. In order to clearly describe the AED-LGB method, the following first describes the concepts of autoencoder and LightGBM.

3.1. AutoEncoder

The autoencoder algorithm is a common unsupervised learning dimensionality reduction algorithm based on neural networks. The purpose of an autoencoder is to convert high-dimensional data into low-dimensional data using neural networks, similar to principal component analysis (PCA), but overcoming the linear limitations of PCA. Autoencoders, which have symmetrical network structures, consist of two main parts [27]: the encoder and the decoder. The encoder compresses the original high-dimensional input data, while the decoder reconstructs the original high-dimensional input data. The overall structure is shown in the following Figure 1.

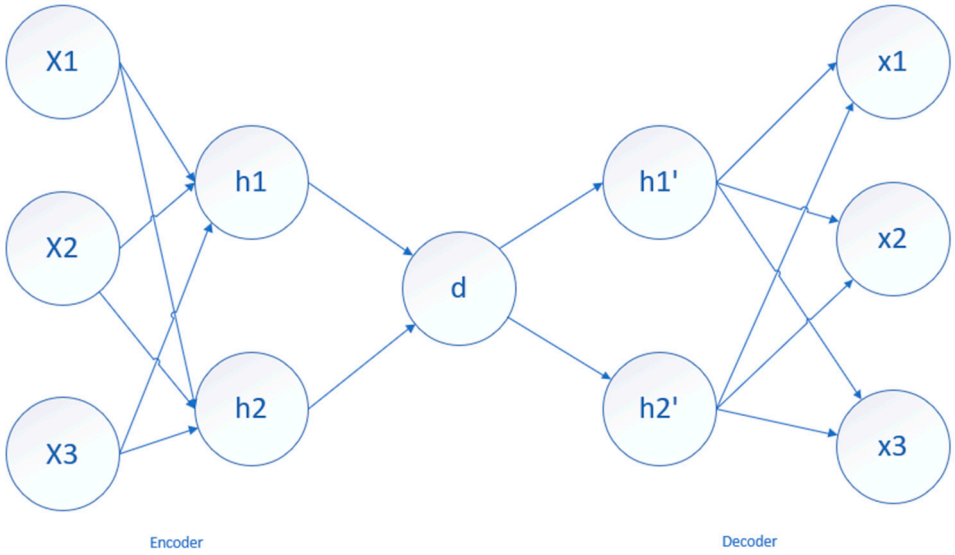

Figure 1. Typical AutoEncoder Structure.

The left part is the encoder layer; the encoder architecture is comprised of a series of layers with nodes, which then encode the input into the next layer. The middle part is the latent view representation. The latent view represents the lowest level space in which the inputs are reduced, and information is preserved. The right part is the decoder layer; the decoding architecture is the mirror image of the encoding architecture, and the number of nodes is equal to that of the input. The encoding process of the original data X from the input layer to the hidden layer is the following Equation (1):

$$h = g\theta 1(x) = \sigma(w1x + b1) \tag{1}$$

The decoding process of the original data X from the hidden layer to the output layer is the following Equation (2):

$$\hat{x} = g\theta 2(h) = \sigma(w2h + b2) \tag{2}$$

The difference between the input and the output is called the reconstruction error. The autoencoder model is trained with the goal of minimizing reconstruction errors.

3.2. LightGBM

LightGBM (Light Gradient Boosting Machine) is a framework that implements the GBDT (Gradient Boosting Decision Tree) algorithm [28], which supports efficient parallel training, faster training speed, lower memory consumption, better accuracy, and distributed support for quickly processing massive data. It employs a leaf-wise algorithm with depth limitations instead of the level-wise decision tree growth strategy used by most GBDT tools. This strategy finds the leaf with the highest split gain from all current leaves and then splits it, repeating this cycle. Therefore, compared with level-wise, the advantages of leaf-wise are that it can reduce errors and gain better accuracy under the same number of splits. However, its disadvantage is that it may grow deeper decision trees, resulting in overfitting, as shown in the following Figure 2.

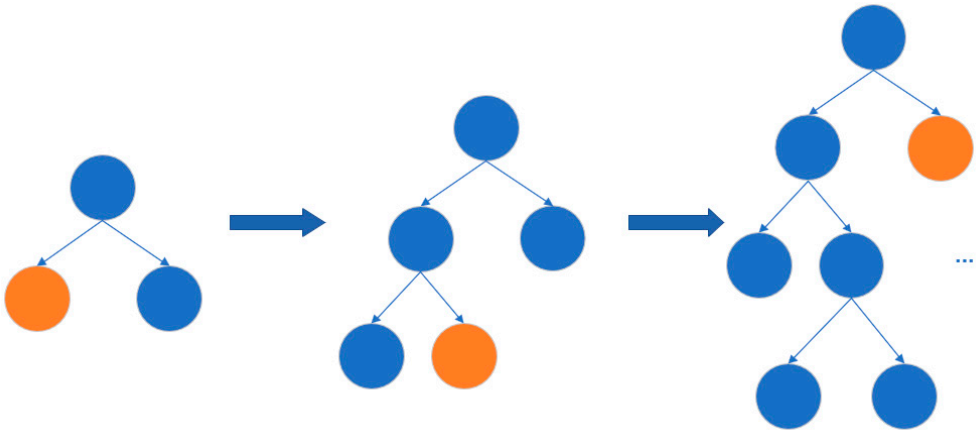

Figure 2. Leaf-wise tree growth.

To avoid over-fitting problems in LightGBM, a histogram-based algorithm and trees' leaf-wise growth strategy with a maximum depth limit can be used to improve the training speed and reduce memory consumption. The hyperparameters that can be tuned for this include "num_leaves", which is the number of leaves per tree, "max_depth", which is the maximum depth of the tree, and "learning_rate", which is used to balance the weight of the class [29]. We need to find a suitable range for this algorithm to obtain better optimization results.

3.3. The AED-LGB Method

Figure 3 is the flowchart of the AED-LGB method. First, the data is pre-processed, abnormal data is processed, data normalization is performed, and so on. Then, the cleaned data is divided into a training set, a validation set, and a test set. The training set data is input into the autoencoder framework for model training, and during training, the weights and parameters are constantly adjusted through backpropagation to obtain an autoencoder model. Then, the feature coding of dimensionality reduction data extracted by the autoencoder model is put into the LightGBM network for training, and a certain output probability is used to determine whether it is fraudulent or normal data. When training the LightGBM model, we also need to determine the appropriate classification probability threshold to obtain better performance. To verify the models' effectiveness, the autoencoder and LightGBM models, along with the threshold value that has been determined, can be applied to each test data point. This will allow us to determine whether each test data point is categorized as fraudulent or normal based on the trained models and threshold.

In the AED-LGB method, the LightGBM model with probabilistic classifications is used to classify a datum as fraudulent with probability p and as normal with probability $1 - p$, where $0 \leq p \leq 1$. Subsequently, AED-LGB outputs the final classification as fraudulent if p is greater than a predetermined classification probability threshold θ. Therefore, fine-tuning the probability threshold value θ can provide a customized classification result for AED-LGB.

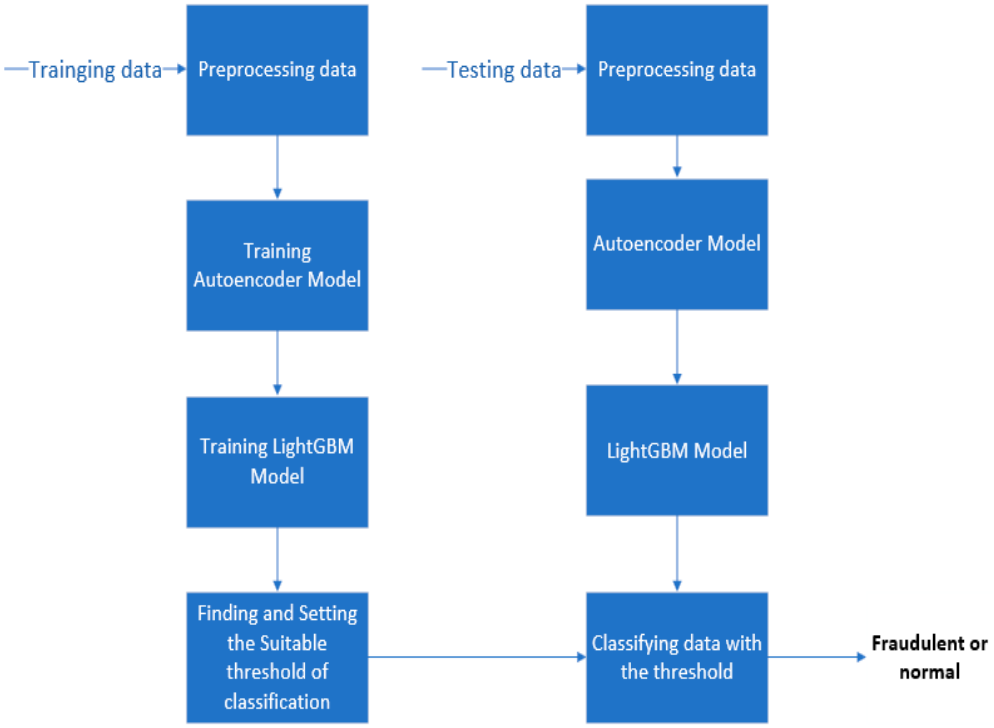

Figure 3. The flowchart of the AED-LGB method.

The pseudocode of the proposed AED-LGB method is shown as Algorithm 1 below.

Algorithm 1: AED-LGB Algorithm

Input: Xtrain: trainging data; Xvalidation: validation data; Xtest: test data;
Output: the classification result for every test data, the result is 0 or 1, 0 represents normal and 1 represents fraudulent;

```
Train the autoencoder model AEMT with Xtrain
MT ← AEMT(Xtrain)
Train the Lgbm model LGT with MT
MV ← AEMT(Xvalidation)
for θ ← 0 to 1 step 0.01
    for each v in MV do
    p ← LGT(v)
    if p > θ then
    result[θ][v] ← 1
    else
    result[θ][v] ← 0
    end if
    end for
end for
Find the best θ in terms of metric
MC ← AEMT(Xtest)
for each c in MC do
    q ← LGT(c)
    If q > θ then
        ouput[c] ← 1
    end if
end for
return ouput
```

The pseudocode is summarized in the following steps:

Step 1: Train the autoencoder model AEMT with the training dataset, and gain a training dataset featuring code set MT.

Step 2: Train the LightGBM model LGT with the training dataset feature code set MT generated in Step 1.

Step 3: Apply the AEMT model to the validation dataset and extract the validation dataset feature code set MV.

Step 4: For threshold θ = 0 to 1 steps (=0.01), at each step, perform the following operation: input each code in MV into the LGT model, gain a probability value *p*. When *p* is greater than the threshold value, the result is 1, which is fraudulent data; otherwise, it is 0, which is normal data.

Step 5: Use all the classification results in Step 4 to find the best threshold according to the metric to produce the best classification performance.

Step 6: Apply the AEMT model to the test dataset and extract the test dataset feature code set MC.

Step 7: Input each codeC in MC into the LGT model to obtain a probability value q, compare q with the best threshold determined in Step 5, and finally, output the classification result.

4. Discussion and Experimental Results

In order to evaluate the effectiveness of our method, we conducted our validation on a credit card transaction dataset. This dataset presents transactions that occurred in two days, where we have 492 frauds out of 284,807 transactions. The dataset is highly unbalanced; the positive class (frauds) accounts for only 0.172% of all transactions.

Before modeling the data, we pre-processed the data and removed abnormal points. At the same time, due to the data imbalance, when predicting the model, it may not be able to make the right prediction, and the final model will tend to predict the majority dataset, and the minority will definitely be ignored. To solve this problem, we can use undersampling and oversampling techniques. Undersampling will lose a large amount of data, which will reduce the number of training data samples and affect the training effect of the model. Oversampling technology duplicates the minority class samples multiple times, enlarges the data scale, and increases the complexity of model training, but at the same time, it very easily causes overfitting. Here we use a type of oversampling technology smote algorithm [30]. The smote algorithm for each sample x in the minority class randomly selected one sample y from its k-nearest neighbors and then randomly selected a point on the x, y line as the new synthetic sample. This kind of new synthetic sample oversampling method can reduce the risk of overfitting.

4.1. Data Pre-Processing

Before training a model, we need to perform data pre-processing, which is an essential step. We need to clean the data, select the features that affect the model training, and normalize the features.

Table 1 shows some of the features of the dataset used in this experiment. The features of v0–v28 have been anonymized, and their feature values are the features after PCA. Here we only show six features related to v. The class feature is the label feature; 1 represents a fraudulent transaction, and 0 represents a normal transaction. The Amount feature is the transaction amount. Since the mean and variance of the Amount feature are significantly different from those of the anonymous features v1–v28, normalization can be performed on the amount. The time feature is the time interval between all transactions and the first transaction. Since no time-sequence-related analysis or modeling was performed in this experiment, it is removed.

We performed data analysis on the class label histogram. As shown in Figure 4, the data is imbalanced, and if the imbalanced data is directly used for training, it will often lead to inaccurate predictions.

Table 1. Some features of the dataset from a bank.

Time	V1	V2	V3	V4	V5	V28	Amount	Class
0	−1.359807134	−0.072781173	2.536346738	1.378155224	−0.33832077	−0.021053053	149.62	0
406	−2.312226542	1.951992011	−1.609850732	3.997905588	−0.522187865	−0.143275875	0	1
472	−3.043540624	−3.157307121	1.08846278	2.288643618	1.35980513	0.035764225	529	1

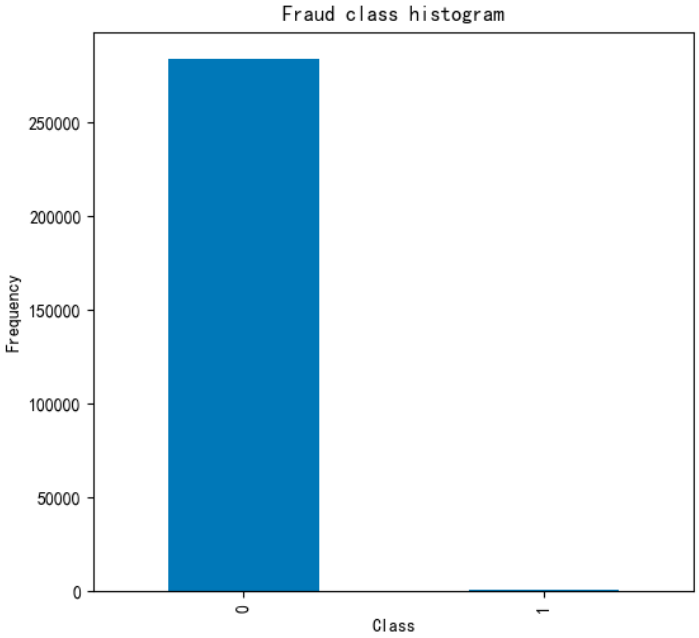

Figure 4. Fraud class histogram before oversampling.

The number of sample data for 0 is much larger than that for 1. To balance the two sampled data, we introduce the smote over-sampling algorithm, which increases the amount of sampled data for class 1, as shown in Figure 5.

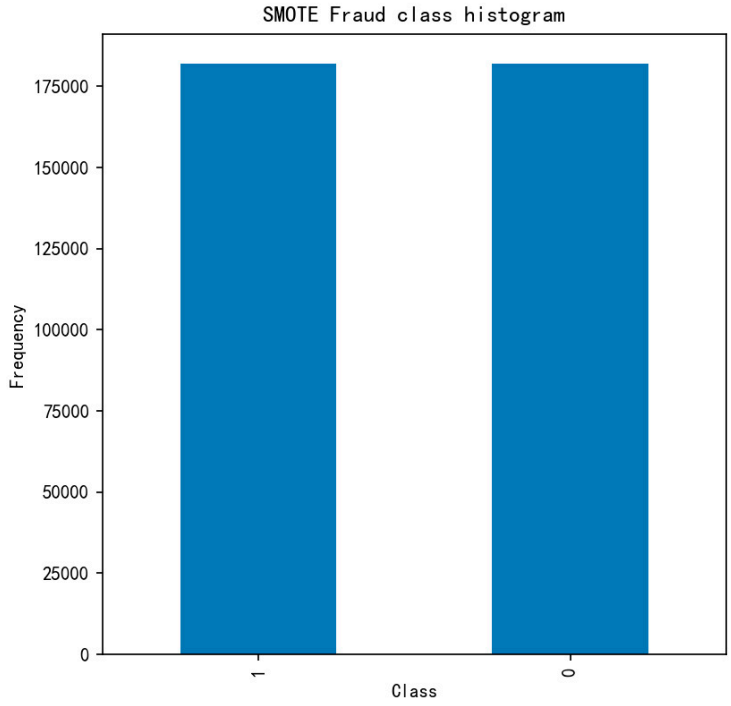

Figure 5. Fraud class histogram after oversampling.

After oversampling, the number of class 1 and class 0 is obviously the same. Here, we keep the data set before and after data enhancement to prove whether the AED-LGB algorithm can also have a strong performance without data enhancement.

4.2. Experiment and Performance Evaluation of the AED-LGB Algorithm

The following is the part of designing and verifying the idea of the AED-LGB algorithm we proposed. We will describe how to design the AED-LGB algorithm in detail and draw conclusions through comparative experiments.

After data pre-processing, We use stratified sampling to divide the dataset into training, validation, and test sets in a 6:2:2 ratio. Then we conduct experiments according to the algorithm flow mentioned above.

In our experiment, for the network architecture design of the autoencoder model, a total of seven layers of the neural network were designed. As shown in Figure 6, we need to pay attention to the relevant dense information in the figure. The output of the previous layer's neural network is connected to the input of the next layer's neural network, and the number of neurons in both the input and output are the same. For example, in the figure, the first dense input is 30, and the output is 256, while the input of the next layer is 256, and the output is 128.

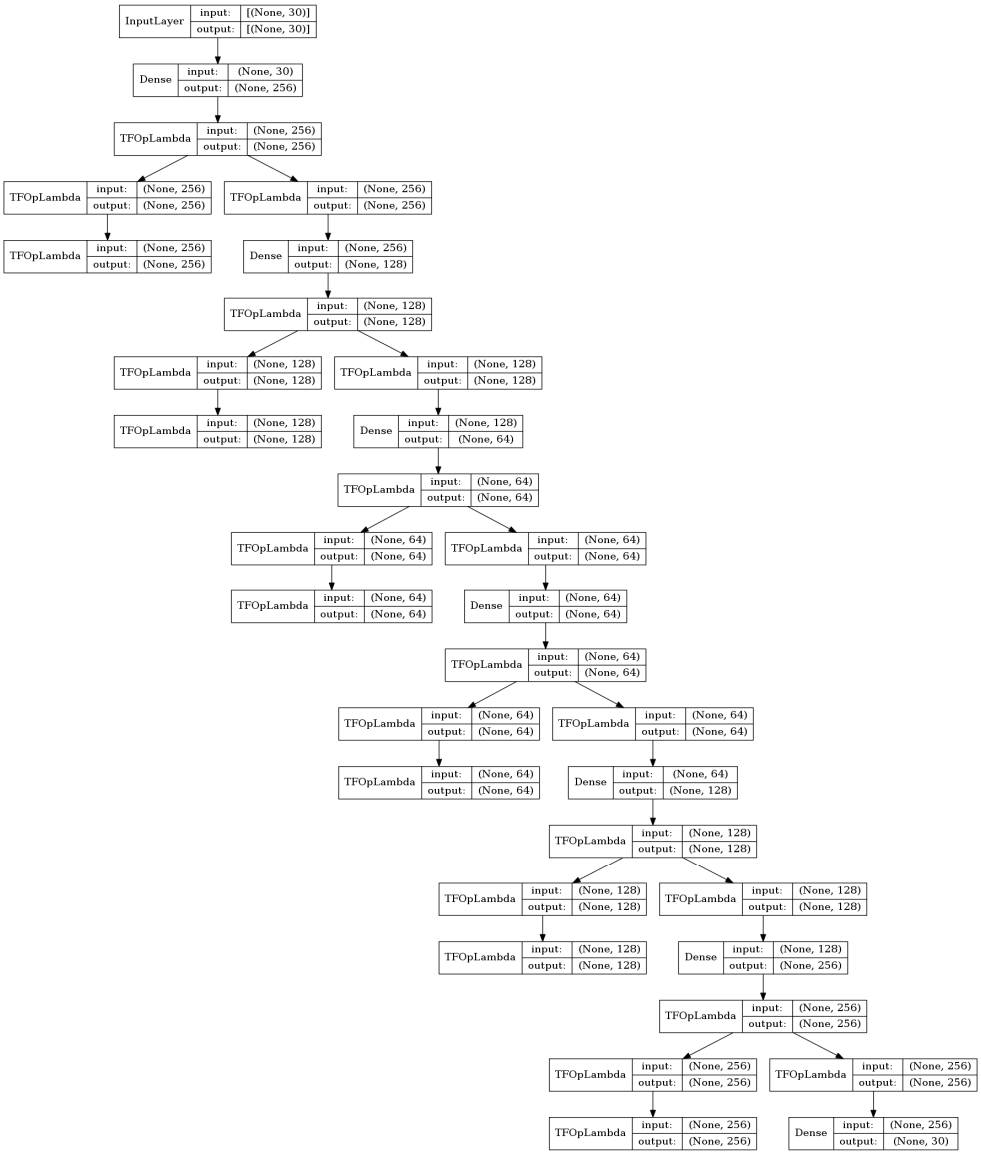

Figure 6. The network architecture design of the autoencoder.

The activation function uses the new deep learning activation function Mish, which is based on Diganta Misra's paper "Mish: A Self Regularized Non-Monotonic Neural Activation Function". Mish is superior to ReLU at high-significance levels ($p < 0.0001$) [31]. The autoencoder model uses adam as the optimizer. To prevent overfitting, L2 regularization is applied to every layer of the encoder and decoder. At the training stage, early stopping is adopted to prevent overfitting, using validation loss as the monitor. The loss function adopts MSE. The LGB model (LightGBM) sets the maximum depth to four, the learning rate to 0.05, and the number of leaf nodes to seven. It generates probabilistic classification and classifies the test data as fraudulent with probability p, $0 \leq p \leq 1$. From the above text, we used the smote algorithm to perform data resampling. The performance evaluation of the AED-LGB algorithm is divided into two experiments. We will conduct the experiment on the data before and after sampling and then examine the performance index evaluation comparison. First, we experiment with the data before sampling. We train the model; Figure 7 is the comparison of the test and training losses in the training process.

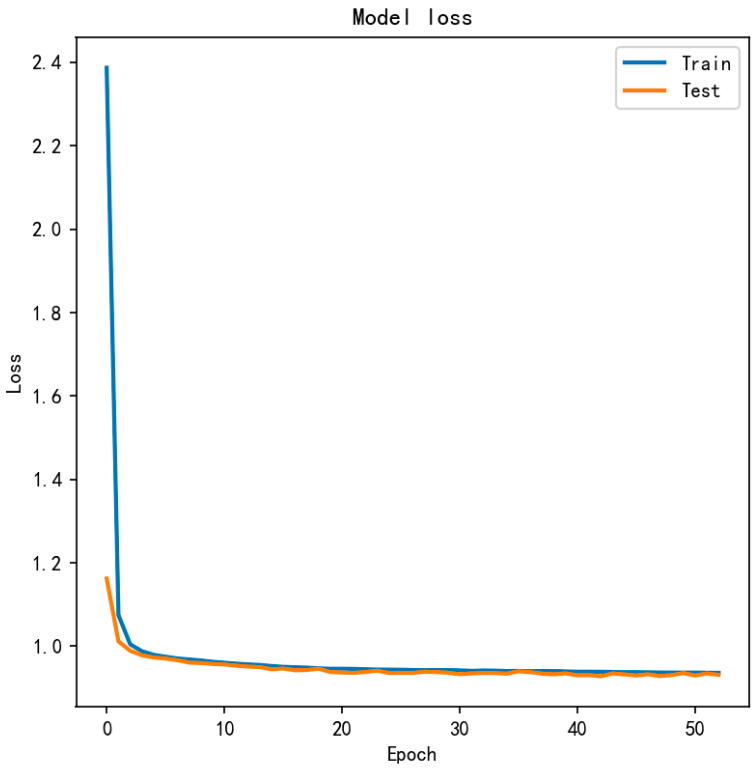

Figure 7. The comparison of the test and training losses in the training process.

It can be seen that the training set and test set are close to each other and nearly coincide after 20 rounds of training, which is a good fit.

LightGBM uses probability classification techniques to check whether test data is classified as fraudulent or not. If p is greater than a pre-set classification threshold θ, it is assumed that the test data is fraudulent. Of course, different threshold values will lead to different classification performances. To find the best threshold to meet different requirements, the threshold needs to be fine-tuned and shifted, and the best threshold under the specific metrics needs to be found. To evaluate the model performance, metrics such as MCC, ACC, TPR, and TNR are employed. The corresponding formulas for the metrics are as follows.

$$TNR = \frac{TN}{TN + FP} \tag{3}$$

$$FPR = \frac{FP}{TN + FP} \tag{4}$$

$$TNR = \frac{TN}{TN + FP} \tag{5}$$

$$TPR = \frac{TP}{TP + FN} \tag{6}$$

$$ACC = \frac{TP + TN}{TP + TN + FP + FN} \tag{7}$$

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP) \times (TP + FN) \times (TN + FP) \times (TN + FN)}} \tag{8}$$

Confusion matrixes are often employed as important indicators in the field of machine learning to evaluate the performance of a model. In Formulas (3)–(6), TP stands for true positive, where the real label is positive, and the model also predicts it as positive, i.e., a true positive. FN: false negative, where the real label is positive, but the model predicts it as negative, i.e., a false negative. FP: false [ositive, where the real label is negative, but the model predicts it as positive, i.e., a false positive. TN: true negative, where the real label is negative, and the model also predicts it as negative, i.e., a true negative. TPR (True Positive Rate), also known as recall (Recall), recall rate, hit rate, or sensitivity (Sensitivity). Recall is for the original positive sample, which can be understood as the number of positive samples that are correctly predicted. TNR (True Negative Rate), also known as specificity (Specificity). Specificity is for the original negative sample, which can be understood as how many negative samples are correctly predicted. FPR (False Positive Rate) can be used as the abscissa of the ROC curve. Formula (8) MCC (Matthew's correlation coefficient), which takes into account the four basic evaluation indicators in the confusion matrix. The MCC describes the correlation between the actual sample and the predicted sample, with a value range of $[-1, 1]$. When the value is 1, it means that the model predicts perfectly; when the value is 0, it means that the prediction result is worse than the random prediction; when the value is -1, it means that the prediction result is extremely poor and almost perfectly avoids the correct answer. In a sense, MCC is comprehensive, and it can be said to be the best metric for binary classification problems [32]. In particular, the two most important metrics are TPR and MCC. The use of TPR as a fraud detection is because the higher the TPR, the more fraud data can be detected, which is the main purpose of fraud detection. However, when evaluating the average performance of the model, MCC will be taken into account, as it takes into account all parameters such as TP, TN, FP, and FN.

101 different threshold values are applied to the AED-LGB classifier. By traversing the threshold values from 0, 0.01…1, we get different ACC, MCC, and other indicators. As is shown in Figures 8–11, ACC values are all as high as above 0.9995, except for 0.0017 when the threshold is 0. The TNR indicator value is 0.999, except for 0 when the threshold is 0. Because the data is unbalanced, the TNR and ACC indicator values are higher, so the MCC indicator is the most important indicator for our experiment. As can be seen from the figure, when the threshold is from 0.2 to 0.8, the overall MCC value is greater than 0.8. From the figure, when the threshold is about 0.3, the MCC value is 0.8478, which is the maximum. However, there is a risk issue if only considering the MCC value. In the credit card anti-fraud problem, another important indicator, TPR, is also meaningful, which is the original positive sample and can be understood as how many positive samples are correctly predicted. Because the higher the TPR value, the more fraudulent data can be detected. When determining the threshold, we can consider the MCC and TPR values at the same time. Combining the MCC curve and TPR curve, it can be seen that threshold 0.2 is more appropriate, both of which values are greater than 0.8. At the same time, we can also consider the threshold value of 0.05 when the MCC value is greater than 0.5 and the TPR is almost 0.9.

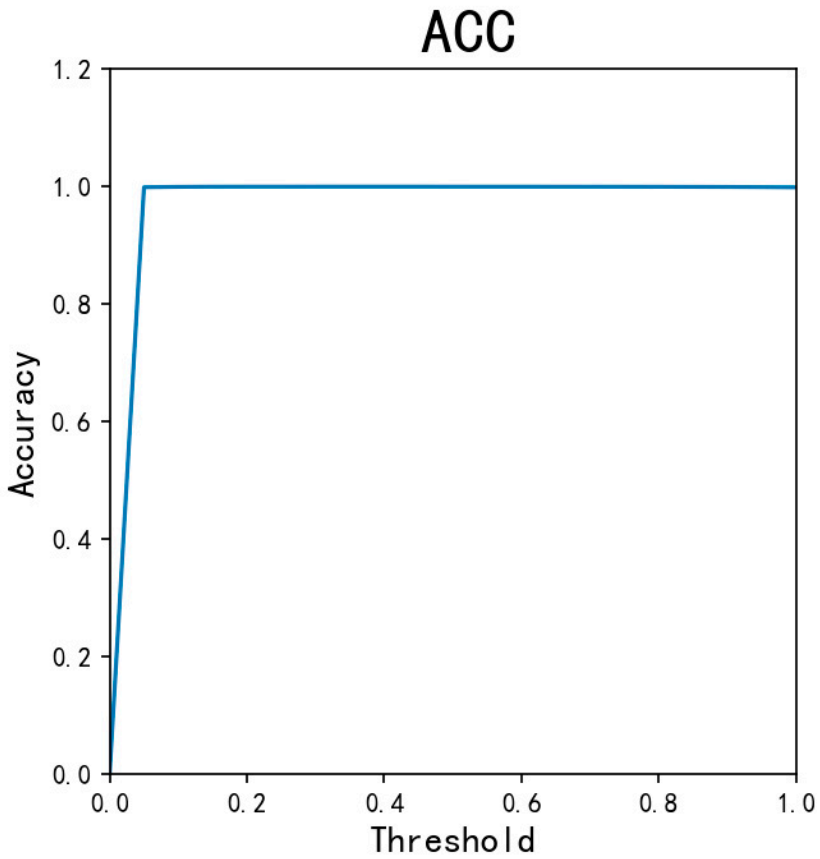

Figure 8. The ACC for different thresholds.

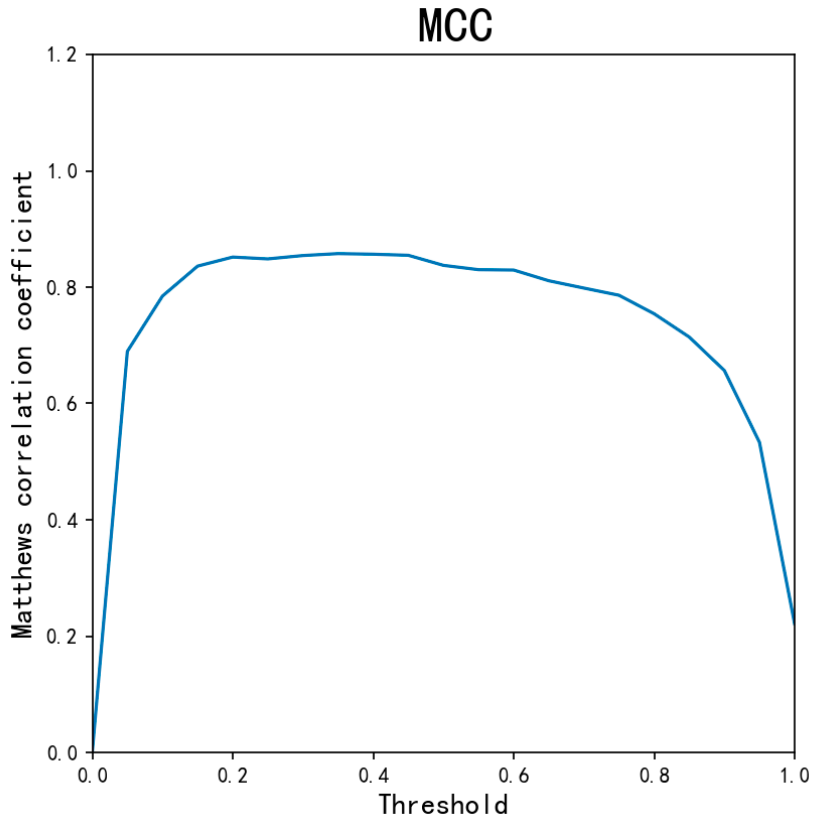

Figure 9. The MCC for different thresholds.

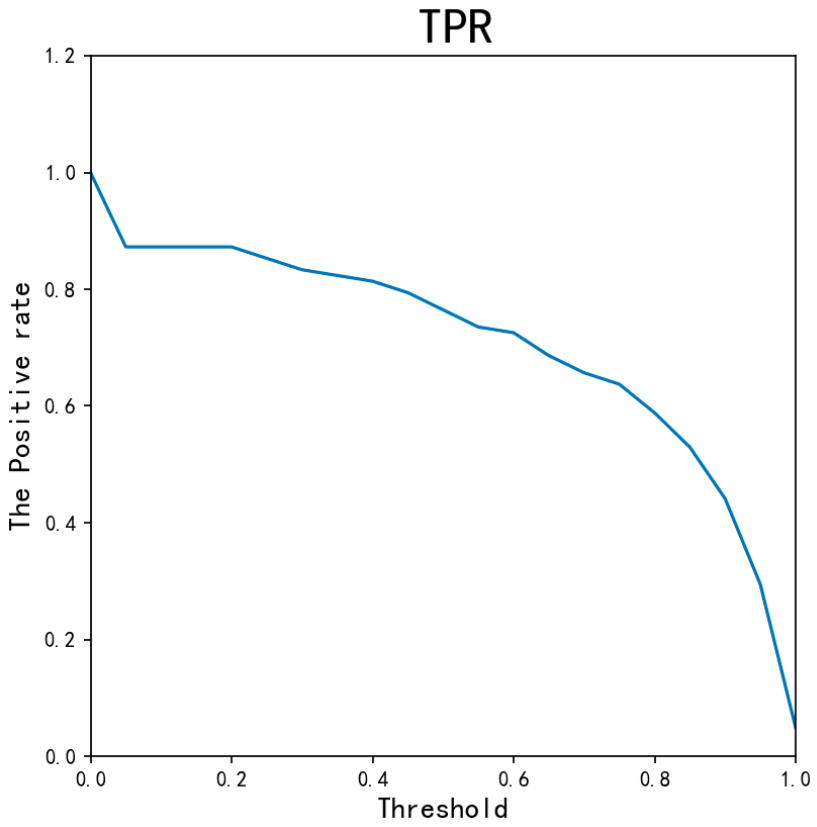

Figure 10. The TPR for different thresholds.

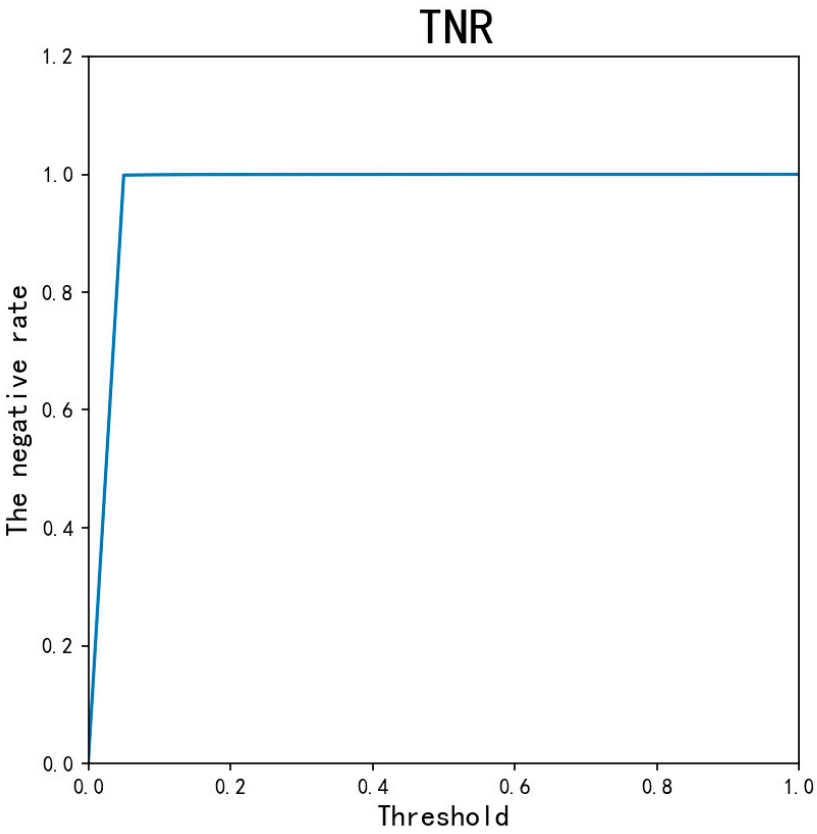

Figure 11. The TNR for different thresholds.

In the second stage of the experiment, we will use the smote algorithm to resample the data to balance it and then repeat the first stage experiment to compare the indicator values with those of the data set without resampling. If it is found that the indicator values of smote algorithm are not improved, it indicates that the AED-LGB algorithm is more suitable for processing unbalanced data sets. Through 100 experiments, after taking the average values of the four indicators, the comparative Table 2 below is obtained by comparing the experimental data after data resampling and before resampling.

Table 2. Performance of AED-LGB with and without data resampling.

Models	ACC	TPR	TNR	MCC
AED-LGB (threshold = 0.2)	0.9993	0.8039	0.9997	0.8506
AED-LGB (threshold = 0.05)	0.9987	0.8929	0.993	0.773
AED-LGB-SMOTE (threshold = 0.70)	0.9970	0.8275	0.9973	0.5574

The AED-LGB-SMOTE algorithm achieves the highest MCC value when the threshold is set to 0.70. It is obvious that the MCC value is lower than that of AED-LGB, and the other indicators ACC and TPR have not been significantly improved. Through the above comparison experiment, it can be concluded that the AED-LGB algorithm is more suitable for the processing of unbalanced datasets.

In addition to the experiments on the AED-LGB algorithm itself in the previous section, in order to further verify the better performance of the AED-LGB algorithm, we can also compare the AED-LGB algorithm with other methods. Here, the commonly used LightGBM, KNN, and random forest algorithms are selected for comparison experiments, and the dataset is still used before the sampling; ACC, TPR, TNR, and MCC are used as performance indicators. The comparison results are shown in the following Table 3.

Table 3. Performance comparisons of AED-LGB and related methods.

Models	ACC	TPR	TNR	MCC
AED-LGB (threshold = 0.2)	0.9993	0.8039	0.9997	0.8506
AED-LGB (threshold = 0.05)	0.9987	0.8929	0.9932	0.773
KNN	0.9691	0.8835	0.9711	0.5903
LightGBM	0.9696	0.8529	0.9995	0.8100
Random Forest	0.9583	0.8025	0.9989	0.6902

From the table, it can be seen that the overall performance of AED-LGB is better than other methods, among which the MCC, ACC, and TNR values of AED-LGB (threshold = 0.2) are the highest, but its TPR value is lower than KNN and LightGBM algorithms. The highest TPR value of AED-LGB (threshold = 0.05) is 0.8929, but its MCC value is lower than AED -LGB (threshold = 0.2). If the TPR value is taken as the most important indicator of credit card fraud, AED-LGB (threshold = 0.05) is the best choice, and its MCC value is also 0.773, which is higher than 0.5, and moreover, is higher than KNN, LightGBM, and random forest models.

4.3. Discussion and Application

From the above experiments, we concluded that the AED-LGB algorithm is more suitable for dealing with imbalanced datasets and shows better performance than traditional machine learning algorithms, such as KNN, LightGBM, and random forest. For example, the AED-LGB algorithm has an overall improvement of 2% in terms of the ACC index compared to LightGBM and KNN. When the threshold is set to 0.2, the MCC index of AED-LGB is 4% higher than that of the second-highest LightGBM algorithm and 30% higher than that of KNN.

Through comparison with the indicators of the LightGBM model, we found that adding the autoencoder improved the learning ability of features.

However, there is a problem of high model training complexity when training the AED-LGB algorithm, and there is still room for improvement in the values of MCC and TPR. We need to try to improve the AED-LGB performance by fine-tuning the hyperparameters of the autoencoder and LightGBM models. The AED-LGB algorithm has already been applied to credit card fraud detection in a certain bank and will be tried in more risk control scenarios in the future.

5. Conclusions

In this paper, an AED-LGB algorithm was proposed to solve the problem of bank credit card fraud. This algorithm first extracts feature data through an autoencoder and then puts the features into the LightGBM algorithm for classification and prediction. In the training process of the algorithm model, we obtained the optimal threshold value by traversing different threshold values and comparing the indicator parameters. Finally, the model marks the data with a value greater than the threshold as fraudulent data. We used a bank's anonymized dataset as a performance evaluation of the AED-LGB algorithm. However, this dataset is unbalanced, and normal data with class 0 far exceeds fraudulent data with class 1. The data was enhanced by oversampling the dataset with the smote algorithm. The experimental results showed that the overall performance of the AED-LGB-SMOTE algorithm did not improve compared to the AED-LGB algorithm, which indicates that the AED-LGB algorithm is more suitable for dealing with this unbalanced data in the bank's fraud business. Subsequently, the AED-LGB algorithm without data enhancement was compared with KNN, Random Forest, and LightGBM algorithms. When the threshold is 0.2, the MCC, TNR, and ACC values of the AED-LGB algorithm are the highest, and when the threshold is 0.05, the TPR value is the highest. To detect more fraudulent data, use the threshold of 0.05. In the future, we will apply the AED-LGB algorithm to other risk control datasets of banks to verify the generalizability of the algorithm and, at the same time, make improvements and optimize the AED-LGB algorithm to obtain better performance.

Author Contributions: Writing—original draft, H.D.; Writing—review & editing, L.L., A.G. and H.W. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Key R&D Program of China, grant number (2019YFB1405802) and the APC was funded by National Key R&D Program of China.

Data Availability Statement: Data sharing is not applicable to this article.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. de Best, R. Credit Card and Debit Card Number in the U.S. 2012–2018. Statista. Available online: https://www.statista.com/statistics/245385/number-of-credit-cards-by-credit-card-type-in-the-united-states/#statisticContainer (accessed on 10 October 2021).
2. Li, M.S.; Yang, D.; Qin, Y.H. Anti-Fraud White Paper of Digital Finance. Available online: https://www.arx.cfa/~/media/45620250D60C4DEFB081322259723D92.ashx (accessed on 31 May 2018).
3. Gangopadhyay, S.; Zhai, A. CGBNet: A Deep Learning Framework for Compost Classification. *IEEE Access* **2022**, *10*, 90068–90078. [CrossRef]
4. Wu, X.; Jiang, G.; Wang, X.; Xie, P.; Li, X. A Multi-Level-Denoising Autoencoder Approach for Wind Turbine Fault Detection. *IEEE Access* **2019**, *8*, 25579–25587. [CrossRef]
5. Taha, A.A.; Malebary, S.J. An intelligent approach to credit card fraud detection using an optimized light gradient boosting machine. *IEEE Access* **2020**, *8*, 25579–25587. [CrossRef]
6. Chen, Y.-R.; Leu, J.-S.; Huang, S.-A.; Wang, J.-T.; Takada, J.-I. Predicting default risk on peer-to-peer lending imbalanced datasets. *IEEE Access* **2021**, *9*, 73103–73109. [CrossRef]
7. Dal Pozzolo, A. Adaptive Machine Learning for Credit Card Fraud Detection. Ph.D. Thesis, Université Libre de Bruxelles, Bruxelles, Belgium, 2015.
8. Lucas, Y.; Portier, P.-E.; Laporte, L.; Calabretto, S.; Caelen, O.; He-Guelton, L.; Granitzer, M. Multiple perspectives HMM-based feature engineering for credit card fraud detection. In Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing, Limassol, Cyprus, 8–12 April 2019; pp. 1359–1361.

9. Awoyemi, J.O.; Adetunmbi, A.O.; Oluwadare, S.A. Credit card fraud detection using machine learning techniques: A comparative analysis. In Proceedings of the 2017 International Conference on Computing Networking and Informatics (ICCNI), Lagos, Nigeria, 29–31 October 2017; pp. 1–9.
10. Zhang, F.; Liu, G.; Li, Z.; Yan, C.; Jiang, C. GMM-based undersampling and its application for credit card fraud detection. In Proceedings of the 2019 International Joint Conference on Neural Networks (IJCNN), Budapest, Hungary, 14–19 July 2019; pp. 1–8.
11. Ahammad, J.; Hossain, N.; Alam, M.S. Credit card fraud detection using data pre-processing on imbalanced data-Both oversampling and undersampling. In Proceedings of the International Conference on Computing Advancements, Dhaka, Bangladesh, 10–12 January 2020; pp. 1–4.
12. Lee, Y.-J.; Yeh, Y.-R.; Wang, Y.-C.F. Anomaly detection via online oversampling principal component analysis. *IEEE Trans. Knowl. Data Eng.* **2012**, *25*, 1460–1470. [CrossRef]
13. Wiese, B.; Omlin, C. *Credit Card Transactions, Fraud Detection, and Machine Learning: Modelling Time with LSTM Recurrent Neural networks*; Springer: Berlin/Heidelberg, Germany, 2009.
14. Jurgovsky, J.; Granitzer, M.; Ziegler, K.; Calabretto, S.; Portier, P.-E.; He-Guelton, L.; Caelen, O. Sequence classification for credit-card fraud detection. *Expert Syst. Appl.* **2018**, *100*, 234–245. [CrossRef]
15. Randhawa, K.; Loo, C.K.; Seera, M.; Lim, C.P.; Nandi, A.K. Credit card fraud detection using AdaBoost and majority voting. *IEEE Access* **2018**, *6*, 14277–14284. [CrossRef]
16. Hsin, Y.-Y.; Dai, T.-S.; Ti, Y.-W.; Huang, M.-C.; Chiang, T.-H.; Liu, L.-C. Feature engineering and resampling strategies for fund transfer fraud with limited transaction data and a time-inhomogeneous modi operandi. *IEEE Access* **2022**, *10*, 86101–86116. [CrossRef]
17. Naveen, P.; Diwan, B. Relative Analysis of ML Algorithm QDA, LR and SVM for Credit Card Fraud Detection Dataset. In Proceedings of the 2020 Fourth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC), Palladam, India, 7–9 October 2020; pp. 976–981.
18. Shirodkar, N.; Mandrekar, P.; Mandrekar, R.S.; Sakhalkar, R.; Kumar, K.C.; Aswale, S. Credit card fraud detection techniques–A survey. In Proceedings of the 2020 International Conference on Emerging Trends in Information Technology and Engineering (ic-ETITE), Vellore, India, 24–25 February 2020; pp. 1–7.
19. Malini, N.; Pushpa, M. Analysis on credit card fraud identification techniques based on KNN and outlier detection. In Proceedings of the 2017 Third International Conference on Advances in Electrical, Electronics, Information, Communication and bio-Informatics (AEEICB), Chennai, India, 27–28 February 2017; pp. 255–258.
20. Pumsirirat, A.; Liu, Y. Credit card fraud detection using deep learning based on auto-encoder and restricted boltzmann machine. *Int. J. Adv. Comput. Sci. Appl.* **2018**, *9*, 18–25. [CrossRef]
21. Zamini, M.; Montazer, G. Credit card fraud detection using autoencoder based clustering. In Proceedings of the 2018 9th International Symposium on Telecommunications (IST), Tehran, Iran, 17–19 December 2018; pp. 486–491.
22. Krishna, M.V.; Praveenchandar, J. Comparative Analysis of Credit Card Fraud Detection using Logistic regression with Random Forest towards an Increase in Accuracy of Prediction. In Proceedings of the 2022 International Conference on Edge Computing and Applications (ICECAA), Tamilnadu, India, 13–15 October 2022; pp. 1097–1101.
23. Wulsin, D.; Blanco, J.; Mani, R.; Litt, B. Semi-supervised anomaly detection for EEG waveforms using deep belief nets. In Proceedings of the 2010 Ninth International Conference on Machine Learning and Applications, Washington, DC, USA, 12–14 December 2010; pp. 436–441.
24. Zhou, C.; Paffenroth, R.C. Anomaly detection with robust deep autoencoders. In Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, NS, Canada, 13–17 August 2017; pp. 665–674.
25. Chalapathy, R.; Menon, A.K.; Chawla, S. Anomaly detection using one-class neural networks. *arXiv* **2018**, arXiv:1802.06360.
26. Wasikowski, M.; Chen, X.-w. Combating the small sample class imbalance problem using feature selection. *IEEE Trans. Knowl. Data Eng.* **2009**, *22*, 1388–1400. [CrossRef]
27. Wang, M.; Zhao, M.; Chen, J.; Rahardja, S. Nonlinear unmixing of hyperspectral data via deep autoencoder networks. *IEEE Geosci. Remote Sens. Lett.* **2019**, *16*, 1467–1471. [CrossRef]
28. Liang, W.; Luo, S.; Zhao, G.; Wu, H. Predicting hard rock pillar stability using GBDT, XGBoost, and LightGBM algorithms. *Mathematics* **2020**, *8*, 765. [CrossRef]
29. Hashemi, S.K.; Mirtaheri, S.L.; Greco, S. Fraud Detection in Banking Data by Machine Learning Techniques. *IEEE Access* **2022**, *11*, 3034. [CrossRef]
30. Camacho, L.; Douzas, G.; Bacao, F. Geometric SMOTE for regression. *Expert Syst. Appl.* **2022**, *2022*, 116387. [CrossRef]
31. Misra, D. Mish: A self regularized non-monotonic activation function. *arXiv* **2019**, arXiv:1908.08681.
32. Zhu, Q. On the performance of Matthews correlation coefficient (MCC) for imbalanced dataset. *Pattern Recognit. Lett.* **2020**, *136*, 71–80. [CrossRef]