LightGBM Model for Detecting Fraud in Online Financial Transactions

Jipeng Chen*

School of Mathematics, South China University of Technology, Guangzhou, China, 510641

* Corresponding Author Email: 15119953022@163.com

Abstract. With the rapid development of online finance, combined with its significant potential threats, nowadays it has increasingly drawn attention to the issue of fraud in online payment transactions. Due to the large and diverse nature of online payment data and the binary nature of fraud detection, this study implements the machine learning algorithm LightGBM to detect fraudulent activities. Through data exploration and data preprocessing, the data is refined. Leveraging violin plots to observe the distribution of variables in fraudulent and non-fraudulent activities, and then an insignificant variable is removed. Next, the processed data is fed into the LightGBM model. The result yields a fraud prediction accuracy of 99.5%, shows a very high precision. Additionally, the study verifies the algorithm's robustness and generalization ability. Overall, this research holds significant importance, given its findings and implications for addressing fraud in the rapidly evolving landscape of online financial transactions.

Keywords: Fraud detection, one-hot encoding, violin plot, LightGBM, softmax loss function.

1. Introduction

In recent years, the rapid development of modern information technology and globalization has accelerated the transformation of online payment methods in the financial sector. Concurrently, the prevalence of fraudulent transactions has increased, emerging as a pressing issue for banks and the financial industry [1]. Anti-fraud research is crucial for safeguarding user assets and preventing financial crises. The characteristics of digital payment transaction data, characterized by massive volume, diverse sources, and heterogeneity, make the integration of intelligent algorithms based on big data and machine learning particularly relevant for fraud detection [2].

The detection of fraudulent behavior itself constitutes a binary classification problem. In this study, I will select a suitable machine learning algorithm capable of detecting the presence of fraud in transactional activities with high accuracy. This holds significant theoretical research significance for integrating intelligent algorithms into fraud detection in the context of the vast and varied landscape of digital payment transactions. Detecting fraud with high precision is paramount in the realm of online financial payments, as it enables better discrimination of transactional behaviors, thereby enhancing the efficiency of financial services. It's one of the innovative aspects of my paper. In addition, this thesis will employ the observation of variable distributions for variable selection, which constitutes another innovative aspect of this study.

2. Data description

The data set used in this paper is sourced from Synthetic Financial Datasets For Fraud Detection (kaggle.com),consisting of 6,362,620 samples containing 11 variables, including *step, type, amount, nameOrig, oldbalanceOrg, newbalanceOrig, nameDest, oldbalanceDest, newbalanceDest, isFraud, isFlaggedFraud*. The specific explanations of each variable are shown in the following table 1:

Table 1. Specific explanations of variables

Name	Explanation
step	Mapping a unit of time to the real world, where 1 step corresponds to a duration of 1 hour
type	Contain five categories: CASH-IN, CASH-OUT, DEBIT, PAYMENT and TRANSFER
amount	The transaction value in the currency of the local region
nameOrig	The individual initiating the transaction
oldbalanceOrg	The balance prior to the transaction taking place
newbalanceOrig	The updated balance following the completion of the transaction
nameDest	Customer who is the recipient of the transaction
oldbalanceDest	Initial balance recipient before the transaction
newbalanceDest	New balance recipient after the transaction
isFraud	This is the transactions made by the fraudulent agents inside the simulation
isFlaggedFraud	A prohibited action within this dataset refers to any effort to transfer an amount exceeding 200,000 in a solitary transaction

3. Data exploration

Due to the large sample size and high exploratory potential of the data, exploratory data analysis will be conducted below. The descriptive statistics of the data below will lay the foundation for subsequent modeling.

3.1. Exploration of transaction type data

Transaction types can be categorized into five types, including *CASH_IN*, *CASH_OUT*, *PAYMENT*, *DEBIT TRANSFER*. Statistical analysis for these five types is presented in the figure 1.

From the figure 1, it can be observed that the *DEBIT* transaction type has the least amount of data and significantly differs from the data volumes of other transaction types. The transaction types with higher data volumes are *CASH_OUT* and *PAYMENT*.

3.2. Explore the correlation among variables

To investigate the potential correlations among various variables, a correlation analysis was conducted on *step*, *amount*, *oldbalanceOrg*, *newbalanceOrig*, *oldbalanceDest*, *and newbalanceDest* using the Pearson correlation coefficient. The results were visualized in the figure 2.

From the figure, it is evident that there is a strong correlation between *oldbalanceOrg* and *newbalanceOrig*, as well as between *oldbalanceDest* and *newbalanceDest*. The correlations among the other variables are not significant.

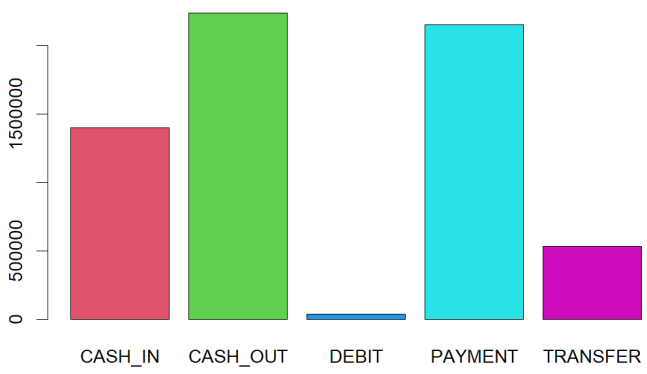

Figure 1. Transaction type statistics

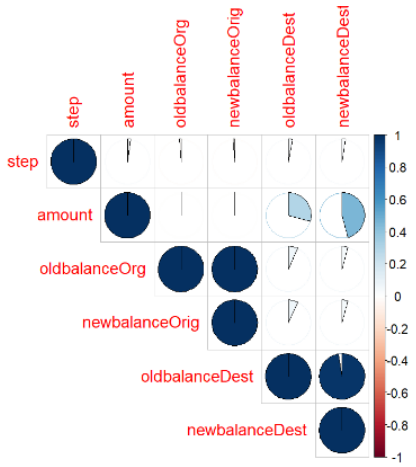

Figure 2. The correlation of each variable

4. Methods

4.1. Data preprocessing

4.1.1 One-hot encoding

The dataset includes a feature variable *type* with five categories: *CASH_IN, CASH_OUT, TRANSFER, DEBIT, and PAYMENT*. Since machine learning models operate based on distances and similarities between features, direct application of categorical variables is not feasible. To address this, we used one-hot encoding to transform these discrete categories into Euclidean space, facilitating distance calculations [3]. However, one-hot encoding all five values would introduce perfect multicollinearity. To mitigate this, we conducted a test by removing one variable and assessing model performance. Removing the *DEBIT* variable and one-hot encoding the rest yielded the highest model accuracy. Consequently, we applied one-hot encoding to the remaining variables, resulting in new features: *CASH_IN, CASH_OUT, DEBIT, and PAYMENT*.

One-hot encoding allows the transformation of categorical features into discrete data, facilitating more reasonable distance calculations between features. Moreover, it expands the feature space to some extent. In this case, a single feature variable *type* has been expanded into four distinct feature variables through one-hot encoding.

4.1.2 Max-min normalization

Due to the original data having different scales, it is necessary to standardize the data to enhance the reliability of the model. We chose to use max-min normalization, which transforms the data into decimal values within the [0,1] range. After standardization, the feature variables have a certain level of comparability in terms of numerical values, significantly improving the accuracy of the classifier. The specific formula is as follows:

Transforming the feature variables $x_1, x_2, \ldots, x_n$,

$$y_i = \frac{x_i - \min\limits_{1 \le j \le m} x_{ji}}{\max\limits_{1 \le j \le m} x_{ji} - \min\limits_{1 \le j \le m} x_{ji}} \tag{1}$$

The new sequence $y_1, y_2, \ldots, y_n \in [0,1]$ and is dimensionless.

4.1.3 Resampling

Upon statistical analysis of the original data, it is observed that out of 6,362,620 samples, non-fraudulent instances amount to 6,354,407, whereas fraudulent instances are only 8,213. The sample ratio is 773:1, as illustrated in the figure 3 below:

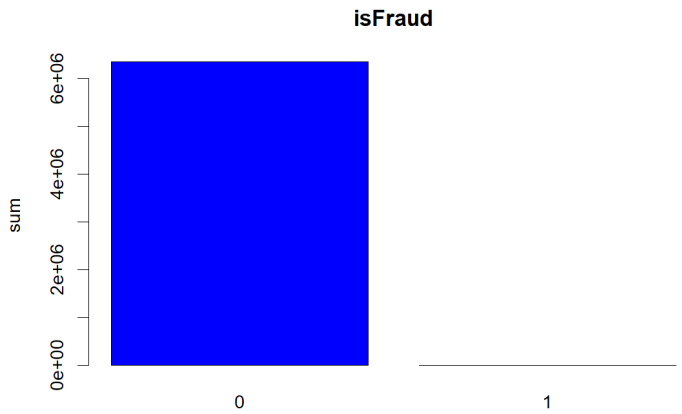

Figure 3. Initial sample size of fraud and non-fraud

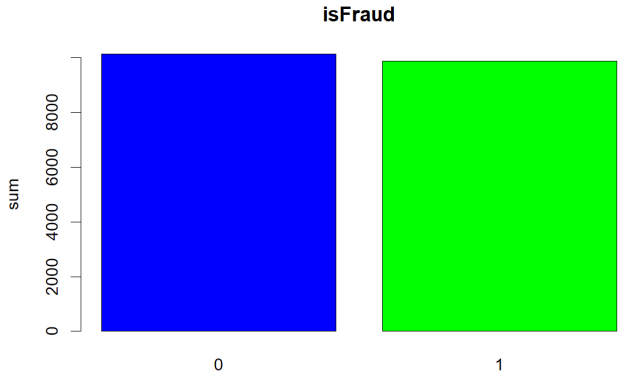

Figure 4. The sample size after sampling

To address extreme imbalance in a binary classification problem, where the model tends to favor non-fraudulent behavior, we aim to balance the classes at a 1:1 ratio. Employing data resampling, we'll use over-sampling for the minority class, increasing its samples beyond the original count [4]-[5]. Then, we'll apply under-sampling to the majority class. As shown in the figure 4, this approach ensures a total sample size of 20,000, with 10,122 samples for non-fraudulent and 9,878 samples for fraudulent behavior, improving classifier generalization and accuracy.

4.2. x-y relationship

To explore the relationship between each variable and our selected dependent variable *is Fraud*, i.e., whether there are significant differences in the distribution of variables between fraudulent and non-fraudulent samples, we have chosen to visualize these differences using violin plots. The details are presented in the following figure 5.

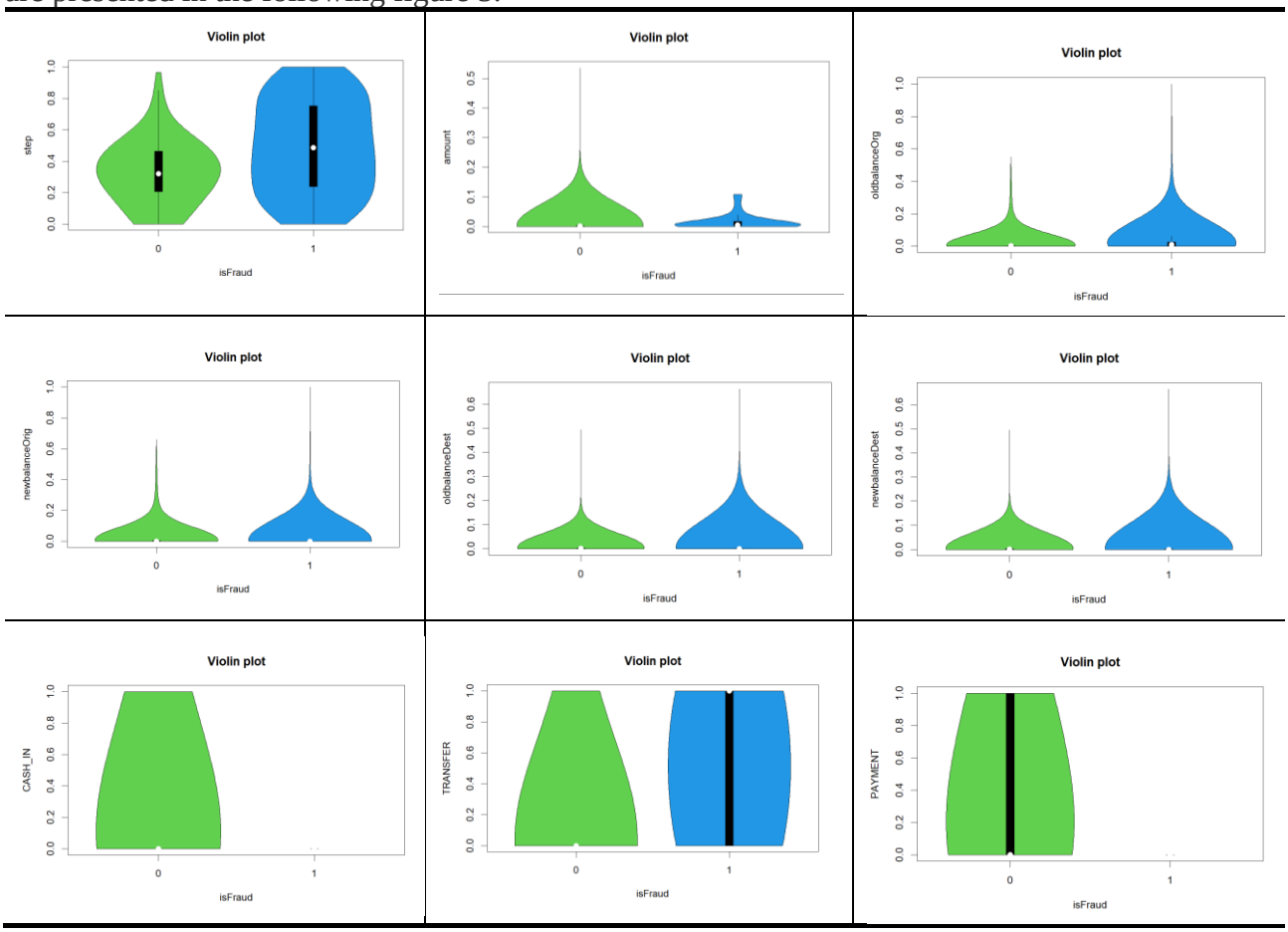

Figure 5. Violin figures of the data distribution of each variable in fraud and non-fraud

From the nine violin plots above, it is evident that for variables such as *step, amount, oldbalanceOrg, newbalanceOrig, oldbalanceDest, newbalanceDest, CASH_IN, TRANSFER, and PAYMENT*, there are significant distribution differences between fraudulent and non-fraudulent samples. Hence, these variables are likely to have a substantial impact on the model. We will retain these variables for subsequent modeling processes.

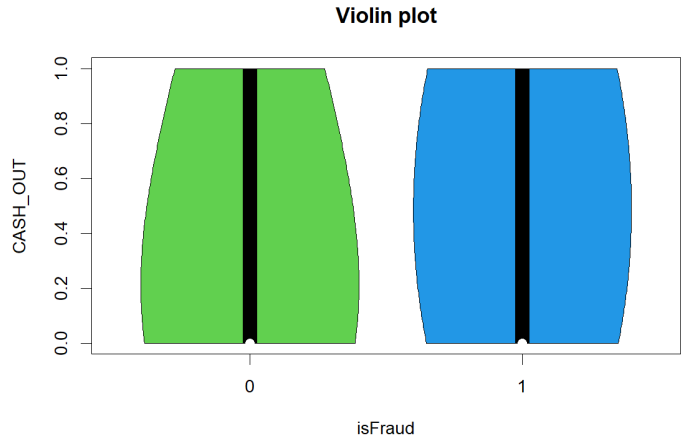

Figure 6. Violin figure of the data distribution of CASH_OUT variable in fraud and non-fraud

As is shown in the figure 6, it is evident that there is minimal difference in the distribution between fraudulent and non-fraudulent samples. Therefore, this variable *CASH_OUT* has little to no impact on the model and will be removed in the subsequent modeling process.

4.3. LightGBM model

LightGBM, an innovative implementation of Gradient Boosting Decision Trees (GBDT), incorporates gradient-based one-sided sampling (GOSS) and exclusive feature bundling (EFB).

4.3.1 Gradient-based One-Side Sampling (GOSS)

Observations reveal that data samples with varying gradient magnitudes exert differing influences on information gain. Notably, samples with larger gradients make a more significant contribution to information gain compared to those with smaller gradients. Thus, the strategy involves preserving samples with substantial gradients while randomly discarding those with minor gradients, effectively reducing memory usage and runtime. The GOSS algorithm implements this strategy by prioritizing samples with large gradients and employing random sampling for samples with smaller gradients. Specifically, GOSS begins by sorting samples based on the absolute values of their gradients and retaining the top (a) samples. Subsequently, for the remaining samples, a random subset of (b) samples is chosen. To address potential impacts on data distribution, GOSS introduces a constant multiplier of ((1-a)/b) when computing information gain. This adjustment effectively modifies the sample weights. [6].

Gradient Boosted Decision Trees (GBDT) use decision trees to learn a function from the input space X^s to the gradient space G .Assuming we have a training set with n independent and identically distributed (iid) instances $\{x_1, \cdots x_n\}$,where each x_i is an s-dimensional vector in the space X^s, In each iteration of gradient boosting, the loss function relative to the negative gradient of the output model is represented as $\{g_1, \cdots g_n\}$. The decision tree model splits each node at the feature with the maximum information gain.

Definition (Variance Gain): Let O be the training dataset at a fixed node of a decision tree. The variance gain for the split feature j at the point d is defined as:

$$V_{j|O}(d) = \frac{1}{n_O}\left(\frac{\left(\sum_{\{x_i \epsilon O: x_{ij} \le d\}} g_i\right)^2}{n^j_{l|O}(d)} + \frac{\left(\sum_{\{x_i \epsilon O: x_{ij} > d\}} g_i\right)^2}{n^j_{r|O}(d)}\right) \quad (2)$$

Where $n_O = \sum I[x_i \epsilon O]$, $n_{l|O}^j(d) = \sum I\left[x_i \epsilon O : x_{ij} \leq d\right]andn_{r|O}^j(d) = \sum I\left[x_i \epsilon O : x_{ij} > d\right]$.

For feature j, the decision tree algorithm selects $d_j^* = argmax_d V_j(d)$ and computes the maximum gain $V_j(d_j^*)$.

In our proposed GOSS algorithm, firstly, we sort the absolute values of gradients of our training instances in descending order. Secondly, we retain the top (a×100%) instances with the larger gradients, forming the instance subset (A). Then, for the remaining ((1-a) ×100%) instances with smaller gradients, we create the complementary set A^C. Furthermore, we randomly sample a subset (B) with (b×100%) instances and its size are b×$|A^C|$.Finally, we split instances based on the estimated variance gain on subsets (A) and (B) [7].

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

Where $A_l = \left\{x_i \epsilon A : x_{ij} \leq d\right\}$, $A_r = \left\{x_i \epsilon A : x_{ij} > d\right\}$, $B_l = \left\{x_i \epsilon B : x_{ij} \leq d\right\}, B_r = \left\{x_i \epsilon B : x_{ij} > d\right\}$, the coefficient ((1-a)/b) is used to normalize the gradients on subset B to the size ofA^C.

So, in GOSS, we use the estimate $\tilde{V}_j(d)$ on a smaller subset of instances instead of calculating the exact $V_j(d)$ over all instances to determine the split points. This significantly reduces the computational cost. Continuing the analysis of the generalization ability of GOSS, we consider the generalization error of GOSS as $\varepsilon_{gen}^{GOSS}(d) = |\tilde{V}_j(d) - V_*(d)|$.This represents the difference between the variance gain computed from GOSS-sampled training instances and the true variance of the base distribution.We have $\varepsilon_{gen}^{GOSS}(d) \leq \left|\tilde{V}_j(d) - V_j(d)\right| + \left|V_j(d) - V_*(d)\right| \triangleq \varepsilon_{GOSS}(d) + \varepsilon_{gen}(d)$.

Therefore, if GOSS approximates accurately, the generalization error of GOSS is close to the generalization error computed from all data instances. On the other hand, sampling can increase the diversity of base learners, which may contribute to improving generalization ability.

4.3.2Exclusive Feature Bundling

In datasets with high dimensionality, it's common for the feature space to exhibit a high degree of sparsity. Within this sparse space, numerous features are often mutually exclusive. One strategy is to group these mutually exclusive features together to create a consolidated feature [8]. By employing a feature scanning technique, we can generate a feature histogram equivalent to that of an individual feature. This modification reduces the complexity of histogram construction from O(#data × #feature) to O(#data × #bundle), where #bundle is considerably smaller than #feature. This leads to a substantial enhancement in the computational efficiency of the algorithm while maintaining the same underlying meaning.

Moreover, we note that numerous features, while not entirely mutually exclusive, seldom exhibit non-zero values simultaneously. If our algorithm permits a minor level of conflict, we can consolidate a smaller number of feature bundles, thereby enhancing computational efficiency even further. Introducing random contamination to a small portion of feature values has at most an impact on training accuracy of $O\left([(1-\gamma)n]^{-2/3}\right)$, where γ is the maximum conflict ratio within each bundle.

The feature bundling algorithm hinges on separating original features from the bundles. When bundling multiple features together, it's crucial to ensure that the values of the original features can be identified within the bundle. Using a histogram algorithm for discretizing continuous values into bins, we add a bias constant to feature values. This ensures that distinct feature values are placed in different bins within the bundle. For instance, if bundling features, A and B, where A's original values range [0,10) and B's range is [0,20), adding a bias constant of 10 to B shifts its range to [10,30). The bundled feature values then span [0,30), facilitating confident merging of A and B.

5. Results

After fitting the data into the basic logistic regression model, I observed an accuracy of only 0.836, which did not meet my expected performance. In this section, we will use the processed data to train and predict with the LightGBM model. The following will mainly introduce the cross-validation, loss function, confusion matrix, and model evaluation applied in the model.

Firstly, LightGBM uses simple cross-validation, splitting the sampled data into 70% for training and 30% for testing based on an empirical rule. The model undergoes training on the training set, and parameters are validated on the test set. This process iterates with new training and test sets, evaluating the model. The optimal model parameters are selected based on a chosen loss function. Cross-validation prevents randomness issues from a single sampling, enhancing the model's generalization.

Secondly, we use the LightGBM model to solve classification problems, and select the objective "multiclass", where the loss function used is the "softmax" function [9]-[10].

The "softmax" function is applied to a vector, denoted as z, which comprises K real numbers. This function normalizes the input vector, transforming it into a probability distribution with K probabilities. The probabilities are proportional to the exponentials of the respective input numbers. Regardless of the initial values being positive, negative, zero, or greater than 1, the "softmax" function scales them to a range between 0 and 1. This enables the interpretation of the transformed values as probabilities. In this paper, a binary classification problem is considered, so k=2. The mathematical formula is:

$$\sigma(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{k} e^{z_j}} \; for\; i = 1, \dots, k \; and \; \boldsymbol{z} = (z_1, \dots, z_k) \epsilon R^k \tag{4}$$

In simpler terms, the "softmax" function works by taking each element in the input vector, raising it to the power of the exponential, and then normalizing the results by dividing them by the sum of all the exponentiated values. This normalization ensures that the sum of the output vector's components equals 1. In essence, the function transforms predicted values, ranging from 0 to 1, into probabilities that are crucial for effective classification. By amplifying the larger values, the "softmax" function enhances decision-making in classification predictions.

Thirdly, using the LightGBM package, the confusion matrix we get is shown in the figure 7:

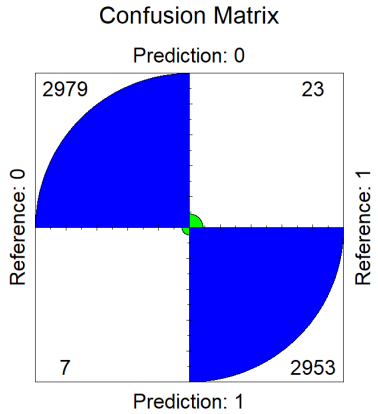

Figure 7. Confusion matrix in the LightGBM model

Since accuracy can be defined as the proportion of correctly classified data instances to the total number of data instances, the accuracy is:

$$Accuracy = \frac{2979 + 2953}{2979 + 2953 + 7 + 23} = 0.995 \tag{5}$$

Lastly, based on the confusion matrix, we calculated that the classification accuracy of the LightGBM model is 0.995, which is more than 10 percentage points higher than the accuracy of 0.836 obtained using the baseline logistic regression model. This indicates that the model we selected significantly improves the classification prediction accuracy. Additionally, we can use the "lgb.importance" function to determine the importance of each feature variable, as shown in the following figure 8.

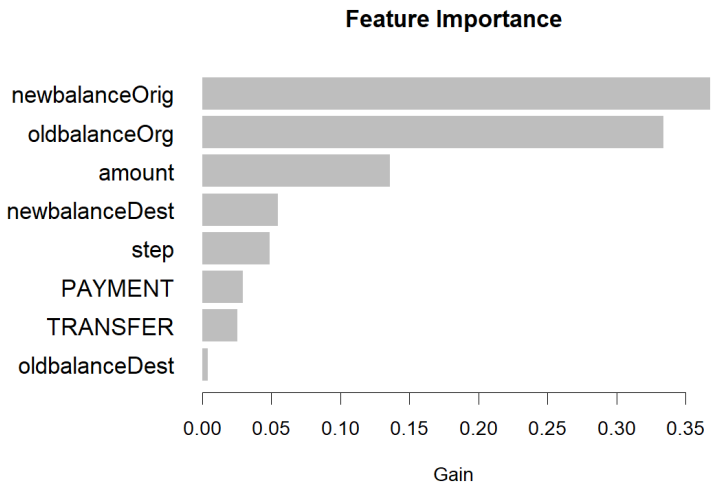

Figure 8. The importance of each variable in the model

In the above figure 8, the horizontal axis represents Gain, which is represents the change in information entropy before and after the split point.

From the figure 8, it can be observed that the scores of the feature variables *newbalanceOrig and oldbalanceOrg* are significantly higher than those of other feature variables. Therefore, these two variables have a substantial impact on the results of classification predictions. The third most important feature variable is *amount*, and the sum of the scores for the top three important feature variables is 0.837. Following these, in terms of importance, are the feature variables *newbalanceDest, step, PAYMENT, TRANSFER and oldbalanceDest.*

Therefore, the LightGBM model we selected achieves a very high accuracy of 0.995 in detecting financial transaction fraud. Moreover, the variables *newbalanceOrig, oldbalanceOrg and amount* are identified as having the most significant impacts on the detection process.

6. Conclusion

In our comprehensive study, involving data exploration, preprocessing, and algorithm implementation, we achieved promising results. Using the LightGBM model on sampled data, we attained a high predictive accuracy of 99.5% in detecting fraudulent behavior. This breakthrough has significant implications for fraud detection. Additionally, our research utilized violin plots to observe variable distributions in fraud and non-fraud activities, serving as a feature selection method. This innovative approach, distinct from model-based variable selection, marks a second noteworthy aspect of our study.

References

[1] LIU Hualing, CAO Shijie, XU Junyi, CHEN Shanghui. Anti-fraud Research Advances on Digital Credit Payment[J]. Journal of Frontiers of Computer Science and Technology, 2023, 17(10): 2300-2324.

[2] C. Wang, C. Wang, H. Zhu and J. Cui, "LAW: Learning Automatic Windows for Online Payment Fraud Detection," in IEEE Transactions on Dependable and Secure Computing, vol. 18, no. 5, pp. 2122-2135, 1 Sept.-Oct. 2021.

[3] Al-Shehari T, Alsowail R A. An insider data leakage detection using one-hot encoding, synthetic minority oversampling and machine learning techniques[J]. Entropy, 2021, 23(10): 1258.

[4] Liu D X, Qiao S J, Zhang Y Q, et al. A survey on data sampling methods in imbalance classification[J]. J Chongqing Univ Technol (NATURAL SCIENCE), 2019, 33: 102-112.

[5] Zhang Jiawei, Guo Linming, and Yang Xiaomei. "Oversampling and Random Forest Improvement Algorithms for Unbalanced Data." Journal of Computer Engineering & Applications 56.11 (2020).

[6] Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., & Liu, T. (2017). LightGBM: A Highly Efficient Gradient Boosting Decision Tree. *Neural Information Processing Systems*.

[7] Pan, H., Li, Z., Tian, C. et al. The LightGBM-based classification algorithm for Chinese characters speech imagery BCI system. Cogn Neurodyn 17, 373–384 (2023).

[8] Wang, Y.; Wang, T. Application of Improved LightGBM Model in Blood Glucose Prediction. Appl. *Sci.* 2020, *10*, 3227.

[9] Franke, M., & Degen, J. (2023, September 28). The softmax function: Properties, motivation, and interpretation.

[10] M. Dukhan and A. Ablavatski, "Two-Pass Softmax Algorithm," *2020 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, New Orleans, LA, USA, 2020, pp. 386-395.