An Optimized LightGBM Model for Fraud Detection

K Huang
University of Illinois at Urbana-Champaign, Champaign, Illinois, 61820, USA
*Corresponding author’s e-mail: khuang966@gmail.com

Abstract. The rapid development of e-commerce and the growing popularity of credit cards have made online transactions smooth and convenient. However, large numbers of online transactions are also the targets of online credit card fraud, which aggregate to enormous losses annually. In response to this trend, many machine learning and deep learning methods have been proposed to solve this problem. Unfortunately, most models have been developed on small datasets and require tedious fine-tuning processes. In this paper, a LightGBM-based method for fraud detection is proposed. The dataset used for this study is the IEEE-CIS Fraud Detection dataset provided by Vesta Corporation, which includes over 1 million samples. Experiments have shown that the LightGBM-based method outperforms most classical methods based on Support Vector Machine, XGBoost, or Random Forest. Besides, effective feature engineering methods for feature selection and Bayesian fine-tuning for automatic hyperparameter searching are also proposed.

1. Introduction
In recent years, credit cards are more and more widely used. According to the Survey of Consumer Payment Choice conducted by the Federal Reserve Bank of Atlanta, 75.5% of U.S. consumers had more than one credit card in 2018[1]. The survey also indicates that most consumers have positive opinions of credit cards because of their convenience. For offline shopping, credit cards simplify the transaction process and save consumers the hassle of carrying cash around and waiting for change. The growing demand for online shopping further promotes the prevalence of credit cards. Many online businesses only accept credit cards or other online payment services based on credit cards. As credit cards grow in popularity, so do fraud targeting credit cards. Fraudsters use various methods to collect or buy credit card information. This information is then used to transfer money from the victim’s account or directly make purchases. Fraudsters usually exhaust the spending power of the credit cards rather quickly to catch the victims off guard. According to the Nilson Report, global losses caused by credit card fraud is expected to exceed $35 billion by 2020[2]. Such losses hurt not only the card holders but also the companies issuing the cards. In response to this trend, financial companies, as well as researchers in this field, are devoting more effort towards effective fraud detection models to decline risky transactions. Many machine learning methods, including logistic regression and SVM, have been proposed in this field. However, since transaction data is difficult to obtain, most of these methods are trained on small datasets and fail to generalize well in production environments. Moreover, heavy reliance on human fine-tuning makes the training process unwieldy and time-consuming.

1.1. Related Work
Machine learning is the process of discovering patterns in training data and using them to make predictions on previously unseen data[3]. For fraud detection, the purpose of the model is to identify
risky transactions based on transaction and identity information. To solve this problem, Ravi and Singh proposed an Auto-Associative Extreme Learning Factory (AAELF) method for one-class classification problems[4]. Their model consists of three layers: input, hidden, and output. The use of layers requires specifying the number of hidden nodes; different distributions, including Uniform, Normal and Logistic distribution, are used to initialize model weights. The model is trained with negative class data only to minimize the MSE score. The model uses ensemble methods to combine multiple Auto-Associative ELM to form an Auto-Associative Extreme Learning Factor. This improves the model’s stability and generalizability on novel data. The model achieves SOTA performance on both bankruptcy prediction and credit card fraud classification. The main drawback of this model is its reliance on human fine-tuning. Gyamfi and Abdulai explored the accuracy of using various SVM for fraud detection[5]. The accuracy of employing SVM and BPN on fraud detection are both around 85%. Notably, there is a sharp drop from training accuracy to testing accuracy. Accuracy for both models declines by more than 10% during the testing phase, suggesting that the models generalize poorly to novel data.

1.2. Contribution
In this paper, a LightGBM-based method[6] is proposed for fraud detection. The proposed model is trained and tested on the IEEE-CIS dataset, which contains more than 1 million transaction records for about 28 thousand cards[7]. It is by far the largest dataset in the field of fraud detection. The large size of this dataset gives the proposed model a superior generalization capability. Identification and transaction information is included in this dataset. In total, the two sets of information produce 435 features for each transaction. Effective feature engineering techniques are employed on the dataset. These techniques include aggregating features based on transaction time, adding descriptive statistical features such as average transaction amount, decoupling compound features such as user browser version, one-hot encoding categorical features, and feature selection with the rfecv algorithm. These techniques enable the proposed model to identify the most important features in the dataset, which improves model accuracy and prevents overfitting. The proposed method is based on the LightGBM classification model. The Gradient-based One-Side Sampling (GOSS) and Exclusive Feature Bundling (EFB) optimizations give the proposed method an edge over other machine learning algorithms in terms of speed and accuracy in classification. Specifically, the proposed method outperforms logistic regression, random forest, and SVM-based models.

Feature fine-tuning has always been a tedious and time-consuming job in the development of machine learning systems. In the proposed model, Bayesian fine-tuning is employed in the pipeline[8]. Bayesian fine-tuning does not have as many limitations on the target function as other methods such as grid search or random search. The optimization target can be stochastic, non-convex, or even non-continuous. Instead of searching the parameter space randomly or uniformly, Bayesian optimizer uses model accuracy from previous iterations to guide the searching process. This helps the searching process converge faster and achieve better results.

The remainder of this paper is organized as follows: Section 2 explains the proposed feature engineering techniques and data cleaning process. Section 3 presents the LightGBM-based model, Bayesian fine-tuning, and parameters of the LightGBM-based model. In Section 4, experiments are conducted on the LightGBM-based model and other machine learning methods. Finally, Section 5 provides a conclusion.

2. Feature Engineering
The IEEE-CIS dataset was split into a training set and a test set. Each set contains about 500 thousand for over 14 thousand users. Both the training set and the test set include a Transaction Table and an Identity Table. The two tables were explored separately and left joined after feature engineering was completed.
2.1. Transaction Table
The transaction consists of 394 features, including 22 categorical and 372 numeric features. Details are shown in Table 1.

| Variables | Type     | Note                  |
|-----------|----------|-----------------------|
| TransactionID | Number   | Foreign key           |
| isFraud   | Categorical |
| card1-card6 | Categorical |
| addr1-addr2 | Categorical |
| TransactionDT | Date Time | datetime of transaction |
| TransactionAmt | Number   | amount of transaction |
| M1-M9     | Categorical |
| P_email domain | Categorical | purchaser email domain |
| R_email domain | Categorical | receiver email domain |
| dist1-dist2 | Number   |
| C1-C14,D1-D15,V1-V339 | Number   |

As a first step, data cleaning was performed to purge the dataset. Transactions with $0 transferred, or the top 0.1% percent largest amounts were dropped. The many empty entries in the dataset were filled with -1000 to make the table complete. Otherwise, null values would always be categorized into one child in tree models, causing the model to overfit the training data. Feature groups C1~C14, D1~D15, and V1~V339 were noted to contain many features that were closely correlated. Feature selection consisted of dropping features that were correlated with already chosen features by a correlation coefficient larger than 0.9. Fewer features improved the proposed model by reducing the chances of overfitting the training data. From observations, the target column isFraud distributed non-uniformly at different times of the day, so aggregate features were generated. These features were aggregated over certain time periods such as by the hour or day. The time of each transaction is determined by the field TransactionDT. Transaction amounts and transaction numbers were also aggregated by the hour or day. These features were appended to the original data. Finally, descriptive statistical features, such as the overall transaction number and averaged transaction amount, were generated. This provided the proposed model with extra context information on the users’ past transactions, helping it make better-informed decisions.

2.2. Identification Table

| Variables | Type     | Note          |
|-----------|----------|---------------|
| TransactionID | Number   | Foreign Key   |
| DeviceType | Categorical  |
|DeviceInfo | Categorical | Device Details |
| id01-id11 | Number   |
| id12-id38 | Categorical  |

The Identification Table contains 41 features including TransactionID, categorical, and numerical features. TransactionID was used as a foreign key, linking the Transaction Table and Identification Table. Categorical information is generally provided in strings. For example, the DeviceType field contains
three values: Desktop, Mobile, and Other. These categorical features were one-hot encoded to reduce memory use. Each feature can be evaluated individually for correlation. Some fields are compound features which can be divided into multiple features. One example of this is id_31, which is the browser information that contains browser type and browser version. These types of features were split in the process. A normalization on all columns was then performed. To further eliminate redundant features and prevent overfitting, automatic feature selection techniques were used to select important features. Five-fold recursive feature elimination with cross-validation (RFECV) was used with the LightGBM model to select important features automatically.

3. LightGBM-Based Fraud Detection Model

In this section, the LightGBM-based model is introduced along with its training parameters and Bayesian fine-tuning procedures. Compared with other methods such as XGBoost[9] or AdaBoost[10], LightGBM employs many optimizations, such as Gradient-based One-Side Sampling (GOSS) and Exclusive Feature Bundling (EFB). The GOSS strategy primarily chooses data with a high gradient for training, and only samples small portions of data with low gradients as a supplement. This method has been proved to be efficient in speeding up the training process and improving the model’s generalization performance. EFB is used for dimension reduction. When training data have many labels, there tend to be more empty entries. Some exclusive columns can be merged together without information loss. Having fewer features also speeds up the training process. Through these optimizations, LightGBM is able to outperform most other machine learning techniques in speed and accuracy. It also supports parallel training and inference. Thus, LightGBM is suitable for industrial use.

Bayesian optimization was used to search for the best parameters efficiently. Bayesian fine-tuning is used in problems where the maximum is sought for an expensive function:

\[ f: x \rightarrow R \]

\[ x^* = \arg \max f(x), x \in X \]

In this case, x stands for hyperparameter and R stands for model accuracy. The main advantage of Bayesian optimization over convex optimization is that there are fewer restrictions on the underlying function f. It can be stochastic, non-convex, or even non-continuous. It is also faster than a grid search or random search. A sequential model-based optimization algorithm is an effective implementation of Bayesian fine-tuning. It uses a small set of samples in X to initialize a probabilistic regression model M. New locations in the domain X are chosen sequentially by acquisition function S. Usually, there are limitations on how many iterations the algorithm is allowed to run. This is represented with integer T. The process can be summarized as follows:

\[
\text{Input: } f, X, S, M
\]

\[
D \leftarrow \text{InitialSamples}(f, X)
\]

\[
\text{for } i \leftarrow |D| \text{ to } T \text{ do}
\]

\[
p(y|x, D) \leftarrow \text{FitModel}(M, D)
\]

\[
x_i \leftarrow \arg \max_{x \in X} S(x, p(y|x, D))
\]

\[
y_i \leftarrow f(x_i)
\]

\[
D \leftarrow D \cup (x_i, y_i)
\]

end for

It is used to tune key parameters such as learning rate, max tree depth, and leaf number.

### Table 3. LightGBM Parameters Table

| Parameter       | Parameter Description    | Value  |
|-----------------|--------------------------|--------|
| max_depth       | the maximum model depth  | 50     |
| reg_alpha       | learning rate            | 0.01   |
| reg_lambda      | sample rate of rows      | 0.8    |
| num_leaves      | number of leaf nodes     | 381    |
| feature fraction| sample rate of columns   | 0.899  |
| bagging_fraction| sample rate of features  | 0.899  |
The table above shows the parameters of the proposed model determined by Bayesian optimization.

4. Experiments
This section explains the training and validation process of the proposed model on the IEEE-CIS dataset. The environments of the experiments were on Ubuntu 18.04 with a GPU Nvidia P100 with 4 processors. The metric used for evaluating the proposed model was AUC. AUC was calculated based on two fields in the confusion matrix (as shown in Figure 1): True Positive Rate (TPR) and False Positive Rate (FPR). TPR and FPR were calculated based on true positive (TP), true negative (TN), false positive (FP), and false negative (FN):

\[
TPR = \frac{TP}{TP + FN}
\]

\[
FPR = \frac{FP}{FP + TN}
\]

Under different thresholds, TPR and FPR changed respectively. By changing the threshold value, pairs of TPR and FPR could be obtained. Plotting these points in a 2D plane produced the ROC curve, as shown in Figure 2; the area under the ROC curve was defined as the AUC score. The recall (x-axis) and the precision (y-axis) for different thresholds were plotted to produce the P\(\mid\)R curve, as shown in Figure 3.

![Figure 1. Confusion Matrix](image1)

![Figure 2. ROC Curve](image2)
Figure 3. P|R Curve

Ideally, AUC should be 1 with 0% false positive cases and 100% true positive cases. In addition to the AUC score of the proposed model, Table 4 also presents AUC and accuracy scores for two other models: logistic regression and SVM[11].

| Models         | AUC Score | Accuracy |
|----------------|-----------|----------|
| Logistic Regression | 0.908     | 0.925    |
| SVM             | 0.926     | 0.942    |
| LightGBM        | 0.961     | 0.942    |
| LightGBM (Bayesian) | 0.973     | 0.987    |

Table 4 shows that the LightGBM (Bayesian) model outperformed other machine learning models by a wide margin. The AUC score shows a significant difference, indicating that the proposed model outperformed Logistic Regression and SVM by at least 4%. In addition, Bayesian optimization was able to improve AUC by more than 1% while reducing the time and efforts needed for fine-tuning.

5. Conclusion
The challenges posed by the continued rise in credit card fraud have promoted the development of different machine learning models for fraud detection. In this paper, a method that utilizes the LightGBM classification model and Bayesian fine-tuning is proposed. Experiments have been conducted to compare the performances of the proposed model and other machine learning models. The results show that the proposed model outperforms logistic regression and SVM-based models in AUC and accuracy scores, suggesting the effectiveness of the proposed model for credit card fraud detection.

Acknowledgment
I would like to thank Yu Liu for help with experiments and fruitful discussions. In addition, I would like to thank the Kaggle platform for the free resources it provided.

References
[1] Foster, K., Greene, C., Stavins, J. (2019) The 2018 Survey of Consumer Payment Choice. Federal Reserve Bank of Atlanta. Retrieved from https://www.frbatlanta.org/banking-and-payments/consumer-payments/survey-of-consumer-payment-choice/2018-survey.
[2] Robertson, D. (2019) Card Fraud Worldwide 2010-2027. The Nilson Report. Retrieved from https://nilsonreport.com/mention/407/1link/.
[3] Michie, D., Spiegelhalter, D.J., Taylor, C.C., Campbell, J. (1995) Machine learning, neural and statistical classification. Ellis Horwood, Upper Saddle River, NJ, USA.
[4] Ravi, V., Singh, P. (2014) Auto-associative extreme learning factory as a single class classifier. In: 2014 IEEE International Conference on Computational Intelligence and Computing Research. Coimbatore, India.

[5] Gyamfi, N.K., Abdulai, J. (2018) Bank Fraud Detection Using Support Vector Machine. In: 2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference. Vancouver, BC, Canada. pp. 37-41.

[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. In: 2017 Conference on Neural Information Processing Systems. Long Beach, CA, USA. pp. 3149-3157.

[7] Vesta Corporation. (2019) IEEE-CIS Fraud Detection. Retrieved from https://www.kaggle.com/c/ieee-fraud-detection/data.

[8] Snoek, J., Larochelle, H., Adams, R.P. (2012) Practical Bayesian optimization of machine learning algorithms. Advances in Neural Information Processing Systems, 4: 2951-2959.

[9] Chen, T., He, T. (2020) xgboost: eXtreme Gradient Boosting. R Package Version 1.1.1.1.

[10] Rätsch, G., Onoda, T., Müller, K.R. (2001) Soft margins for AdaBoost. Machine Learning, 42(3): 287-320.

[11] Vishwanathan, S.V.M., Murty, M.N. (2002) SSVM: a simple SVM algorithm. In: 2002 International Joint Conference on Neural Networks. Honolulu, HI, USA. pp. 2393-2398.