Making Data Analysis Easier: A Case Study on Credit Card Fraud Detection Based on PyCaret

Chang Huang, Pao-Min Tu(✉), and Chun-You Lin

Dongguan University of Technology, Dongguan, China
daiyea231125@163.com, paomin.tu@dgut.edu.cn, 15821393167@139.com

Abstract. As credit card usage surges globally, associated security challenges, particularly credit card fraud, come into sharp focus. The prevailing method of fraud detection entails employing machine learning algorithms—a skillset necessitating specialized programming and algorithmic training. This research endeavors to mitigate this complexity by harnessing PyCaret—a streamlined data analysis tool—for credit card fraud detection. The study constructed ten distinct machine learning classification models, leveraging Kaggle's credit card transaction dataset, to compare diverse models' performance in fraud detection. Notably, the Random Forest Classifier exhibited superior performance metrics, with an accuracy of 0.9996, an AUC of 0.9439, a recall rate of 0.8022, a precision rate of 0.9423, an F1 score of 0.8654, and an AUPRC of 0.79, thereby indicating commendable performance amid severely imbalanced data. This research further highlights PyCaret's user-friendly programming environment and rich visualization capabilities, achievable with a mere twelve lines of code. This potential for simplicity has significant implications for reducing data analysis barriers for non-technical practitioners while offering preliminary data exploration tools for professional data analysts.

Keywords: data analysis · machine learning · credit card fraud detection · PyCaret · random forest classifier

1 Introduction

In spite of the convenience and deferred payment features offered by credit cards, they often pose security issues, such as fraud and data breaches. To address these problems, many countries have strengthened credit card regulations, and issuing banks have intensified their efforts to detect credit card fraud early in order to minimize losses.

Currently, there are generally three types of data analysis methods used for credit card fraud detection. The first involves data mining methods [1, 2], the second utilizes machine learning algorithms [3–8], and the third employs more complex deep learning algorithms [5, 9, 10]. These three categories of algorithms have found wide application in various credit card-related areas, including risk assessment, anti-fraud measures, and credit scoring. However, deep learning outperforms traditional machine learning

S. H. B. D. M. Zailani et al. (Eds.): ICMSEM 2023, 259, pp. 1203–1211, 2024.
https://doi.org/10.2991/978-94-6463-256-9_122

by alleviating the need for extensive feature design and enhancing modeling capabilities. Nevertheless, it comes with greater requirements for computational resources and training time.

Data analysis requires proficient programming skills and algorithm training. To mitigate the barriers associated with data analysis, this study aims to employ a user-friendly data analysis tool called PyCaret to analyze credit card fraud. Compare to Sklearn, which is designed to provide a consistent and user-friendly API catering to users who require fine-grained control over algorithms and processes, PyCaret simplifies the machine learning process through a high-level API, making it well-suited for rapid prototyping and iterative experiments [11, 12].

In this study, we constructed ten popular machine learning classification models using a credit card transaction dataset obtained from Kaggle. By comparing the performance of these different models in credit card fraud detection, we aimed to identify any anomalous transaction behavior. The study has two main objectives: first, to explore the simplicity of programming machine learning algorithms with PyCaret; and second, to investigate the comprehensive output results offered by PyCaret.

2 Methods

2.1 Dataset

The dataset used in this study is sourced from a European credit card company and consists of transaction data spanning two days in September 2013, which is publicly available on Kaggle (https://www.kaggle.com/mlg-ulb/creditcardfraud). The dataset comprises 284,807 samples and includes 30 features. Features V1 to V28 correspond to principal components derived through PCA, while the remaining features, namely "Time" and "Amount," have not undergone any PCA transformation. The "Time" feature represents the number of seconds elapsed since the first transaction, while the "Amount" feature denotes the transaction amount. The binary "Class" label indicates whether a transaction is fraudulent, with 1 denoting a fraudulent transaction and 0 denoting a legitimate one. There are a total of 492 fraudulent transactions, accounting for 0.172% of the dataset, highlighting a significant class imbalance between the positive (fraudulent) and negative (legitimate) classes.

2.2 Classification Algorithms

The primary objective of machine learning classification algorithms is to identify patterns and rules within training data, facilitating the classification of previously unseen data. Prominent classification algorithms include logistic regression, decision tree, support vector machine, naive bayes, K-nearest neighbors, ridge regression classifier, random forest classifier, gradient boosting classifier, adaptive boosting classifier, and LightGBM. These algorithms employ a range of strategies and optimization techniques during the training process, enabling the selection of a suitable classifier based on the dataset's characteristics and the desired performance criteria.

3 Results

3.1 Comparison of Various Classification Algorithms

We applied ten machine learning classification algorithms, including logistic regression, decision trees, and others, to address the classification problem. By utilizing PyCaret, we accomplished all analyses with a concise twelve lines of code (Appendix 6.1). Our experiment encompassed a range of significant algorithms, each possessing unique strengths suitable for different types of data and problems. Through thorough testing, we observed that each classifier exhibited its own set of strengths and weaknesses.

Among the classifiers, the Random Forest classifier outperformed the others (Table 1). This ensemble learning method combines multiple decision trees to enhance classification performance. The Random Forest classifier achieved an accuracy of 0.9996, an AUC of 0.9439, a recall of 0.8022, a precision of 0.9423, an F1 score of 0.8654, a Kappa coefficient of 0.8652, and an MCC of 0.8686. These results demonstrate high accuracy, good AUC, recall, and precision, as well as a high F1 score, Kappa coefficient, and MCC. Based on these findings, we conclude that the Random Forest classifier delivered the best performance for this particular task.

Random Forest classifier is an ensemble learning algorithm that combines multiple decision trees to perform classification tasks. Each decision tree is built on random subsets of training data and features. The Gini coefficient, also known as Gini impurity, is an important metric used in random forests to measure the purity or impurity of a node. It quantifies the distribution of samples across different classes within a given node.

Table 1. Comparison of ten models

Model[1]	Accuracy	AUC	Recall	Precision	F1	Kappa	MCC
rf	0.9996[2]	0.9439	0.8022	0.9424	0.8654	0.8652	0.8686
gbc	0.9994	0.8584	0.7386	0.8816	0.8009	0.8006	0.8052
ada	0.9993	0.9795	0.7259	0.8491	0.7812	0.7808	0.7840
lr	0.9992	0.9524	0.6623	0.8278	0.7345	0.7341	0.7394
dt	0.9991	0.8628	0.7261	0.7448	0.7331	0.7327	0.7338
ridge	0.9989	0.0000	0.4240	0.8435	0.5635	0.5630	0.5971
knn	0.9984	0.6015	0.0534	0.9250	0.1000	0.0998	0.2167
svm	0.9981	0.0000	0.0152	0.0300	0.0202	0.0199	0.0209
lightgbm	0.9963	0.7223	0.5405	0.2368	0.3257	0.3242	0.3540
nb	0.9934	0.9680	0.6499	0.1588	0.2552	0.2531	0.3191

Note: 1. rf (Random Forest Classifer); gbc (Gradient Boosting Classifier); ada (Ada Boost Classifier);
lr (Logistic Regression); dt (Decision Tree Classifier); ridge (Ridge Classifier);
knn (K Neighbors Classifier); svm (SVM - Linear Kernel);
lightgbm (Light Gradient Boosting Machine); nb (Naive Bayes).
2. The yellow background represents the highest score for this item.

For a node with K classes, denoted as $C_1, C_2, ..., C_K$, and a total of N samples in the node, the Gini coefficient is calculated using the following formula:

$$Gini(p) = 1 - \sum_{k=1}^{K} p_k^2 \tag{1}$$

Here, p_k represents the proportion of samples belonging to class C_k with the node. By computing the squared proportions for each class and summing them up, the result is subtracted from 1 to obtain the Gini coefficient.

A lower Gini coefficient indicates higher purity of the node, implying that samples within the node are more likely to belong to the same class. Random forests classifier utilize the Gini coefficient to construct decision trees that maximize purity for each tree and overall model performance.

The Random Forest classifier can be represented by the following formula:

$$\widehat{Y} = \frac{1}{T} \sum_{t=1}^{T} f_t(X) \tag{2}$$

Here, $\widehat{Y}$ represents the final prediction, T is the number of decision trees, and $f_t(X)$ is the prediction of the t-th decision tree for the input feature X.

Another classifier, for example Ridge regression, is a popular regression analysis method used to handle multicollinearity, where predictors are highly correlated. It addresses the problem of overfitting by introducing a regularization term.

The goal of ridge regression is to minimize the following cost function:

$$J(\theta) = \sum_{i=1}^{m} (y_i - h_\theta(x_i))^2 + \alpha \sum_{j=1}^{n} \theta_j^2 \tag{3}$$

Here, m is the number of samples, n is the number of features, y_i is the target value for the i-th sample, x_i is the feature vector for the i-th sample, θ represents the regression coefficients, and α is the regularization parameter.

PyCaret not only allows for the comparison of multiple models at once, but also enables training and testing of individual models. Taking ridge regression as an example, this study utilized the same parameter settings as shown in Table 1 and accomplished model training and evaluation (excluding visualization) in just five lines of code (Appendix 6.2). The execution results are presented in Table 2.

In conclusion, Random Forest classifier is an ensemble learning algorithm that combines multiple decision trees by introducing randomness. It offers improved accuracy and robustness, making it widely used for classification tasks.

3.2 Visualization

We have visually presented the results of the Random Forest classifier using various visualization techniques. Figure 1 showcases the ROC curve, which demonstrates the classifier's ability to differentiate between positive and negative classes, with an outstanding AUC value of 0.92. Figure 2 depicts the learning curve for both the training

Table 2. Results of ridge regression

Fold[1]	Accuracy	AUC	Recall	Precision	F1	Kappa	MCC
0	0.9990	0.0000[2]	0.4744	0.8605	0.6116	0.6111	0.6385
1	0.9989	0.0000	0.4177	0.8462	0.5593	0.5588	0.5941
2	0.9987	0.0000	0.3544	0.7568	0.4828	0.4822	0.5174
3	0.9989	0.0000	0.4557	0.8372	0.5902	0.5897	0.6172
4	0.9989	0.0000	0.4177	0.9167	0.5739	0.5735	0.6184
Mean	0.9989	0.0000	0.4240	0.8435	0.5635	0.5630	0.5971
Std	0.0001	0.0000	0.0411	0.0514	0.0440	0.0440	0.0423

Note: 1. Five-fold cross-validation

2. Models that do not support the "predict_proba" attribute cannot be used for calculating the "AUC".

and test sets throughout the training process. The accuracy of the training set approaches 100% after 70,000 iterations, while the test set reaches its peak performance after 140,000 iterations, indicating a slight degree of overfitting.

In Fig. 3, the PR curve [13] illustrates the classifier's balanced performance of 0.79 when confronted with an extremely imbalanced dataset, where only 0.172% of the transactions are fraudulent. Figure 4 showcases the influence of discrimination threshold variations on precision, recall, and the F1 score. It is evident that the optimal discrimination threshold is 0.48, underscoring the significance of thresholds in addressing class imbalance issues.

The confusion matrix, presented in Fig. 5, reveals the distribution of predictions, showing 56,856 true negatives, 66 true positives, 32 false negatives, and 8 false positives. Figure 6 provides an overview of the classifier's performance in both the positive and negative classes.

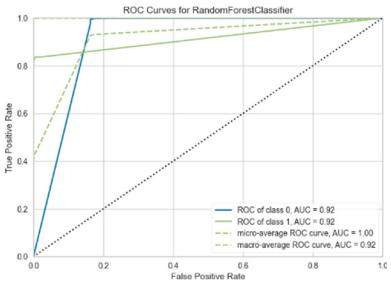

Fig. 1. The ROC curve

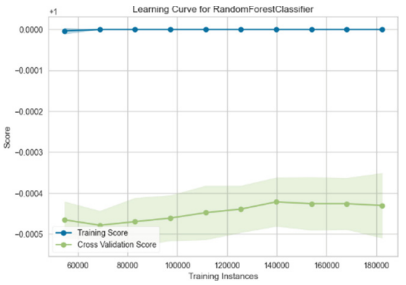

Fig. 2. The learning curve

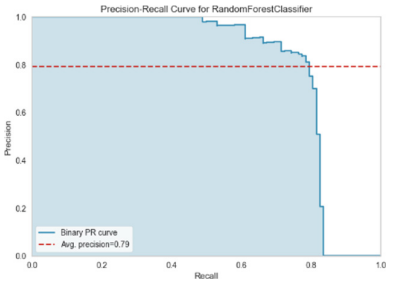

Fig. 3. The PR curve

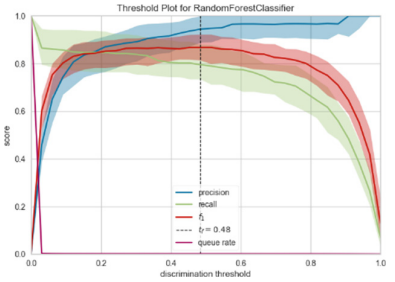

Fig. 4. The discrimination threshold

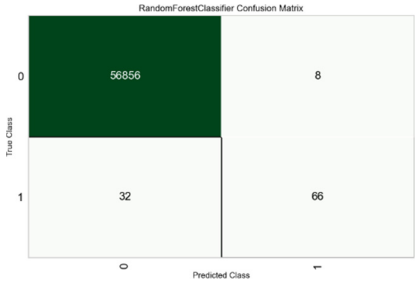

Fig. 5. The confusion matrix

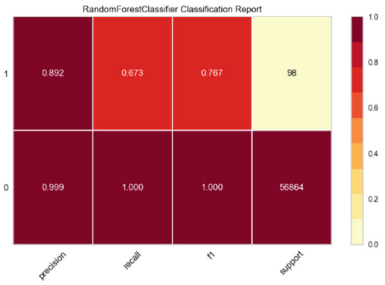

Fig. 6. The classification report

4 Conclusion

We conducted an evaluation of the classification task using ten different machine learning algorithms, and the Random Forest classifier emerged as the top performer. This ensemble learning method, which combines predictions from multiple decision trees, exhibited high accuracy, exceptional classification capability, and effectively balanced precision and recall for positive cases.

Furthermore, this study highlights the user-friendly nature and comprehensive output capabilities of PyCaret. With just twleve lines of code, analysts can seamlessly implement various classification algorithms and generate visualizations, streamlining the data analysis process significantly.

PyCaret serves two main application scenarios. Firstly, it offers a convenient model deployment solution for practitioners without extensive training, lowering the entry barrier for data analysis. Secondly, it provides rapid data exploration tools for professional data analysts. If the results are satisfactory, more complex machine learning frameworks like sklearn can be employed for further optimization. Conversely, if the results fall short of expectations, analysts can explore more intricate deep learning models for further analysis. This approach greatly saves time during preliminary data exploration and allows analysts to allocate more attention to subsequent analysis tasks.

Appendix

Code for Credit Card Fraud Detection Based on PyCaret

```
import pandas as pd
from pycaret.classification import *
# input dataset
df = pd.read_csv('./data/creditcard.csv')
# setup parameters
clf1 = setup(data=df,

target= 'Class',
fold= 5,
session_id= 123,
train_size= 0.8

)

# create and compare allmodels.

best = compare_models()

#create random forest classifier.

rf = create_model('rf')
# Visualization
plot_model(rf, plot='auc')
plot_model(rf, plot='learning')
plot_model(rf, plot='pr')
plot_model(rf, plot='threshold')
plot_model(rf, plot='confusion_matrix')
plot_model(rf, plot='class_report')
```

Code for Ridge Regression Based on PyCaret

```
import pandas as pd
from pycaret.classification import *
# input dataset
df = pd.read_csv('./data/creditcard.csv')
# setup parameters

clf2 = setup(data= df,
target= 'Class',
fold= 5,
session_id= 123,
train_size= 0.8

)

# createridge regression model.

ridge = create_model('ridge')
```

References

1. Beigi, S., Naseri, M.R.A. (2020) Credit card fraud detection using data mining and statistical methods. Shahrood University of Technology, 2. https://doi.org/10.22044/JADM.2019.7506.1894.
2. Bhusari, V., Patil, S. (2011) Application of hidden Markov model in credit card fraud detection. International Journal of Distributed and Parallel Systems, 2(6): 203–211.
3. Maniraj, S.P., Saini, A., Ahmed, S., Sarkar, S.D. (2019) Credit card fraud detection using machine learning and data science. International Journal of Engineering and Technical Research, 08(9). https://doi.org/2019.10.17577/IJERTV8IS090031.
4. Husejinović, A. (2020) Credit card fraud detection using naive Bayesian and C4.5 decision tree classifiers. Periodicals of Engineering and Natural Sciences, 8(1): 1–5.
5. Sharma, P., Banerjee, S., Tiwari, D., Patni, J.C. (2021) Machine learning model for credit card fraud detection- A comparative analysis. The International Arab Journal of Information Technology, 18(6): 789–796.
6. Khatri, S., Arora, A., Agrawal, A.P. (2020) Supervised machine learning algorithms for credit card fraud detection: A comparison. 10th International Conference on Cloud Computing, Data Science & Engineering.
7. Rezapour, M. (2019) Anomaly detection using unsupervised methods: Credit card fraud case study. International Journal of Advanced Computer Science and Applications, 10(11): 1–8.
8. Carcillo, F., Borgne, Y.-A.L., Caelen, O., Kessaci, Y., Oblé, F., Bontempi, G. (2021) Combining unsupervised and supervised learning in credit card fraud detection. Information Sciences, 557: 317–331. https://doi.org/10.1016/j.ins.2019.05.042.
9. Fiore, U., Santis, A.D. Perla, F., Zanetti, P., Palmieri, F. (2019) Using generative adversarial networks for improving classification effectiveness in credit card fraud detection. Information Sciences, 479: 448–455.
10. Murli, D., Jami, S., Jog, D., Nath, S. (2014) Credit card fraud detection using neural networks. International Journal of Students Research in Technology & Management, 2(2): 84–88.
11. Gain, U., Hotti, V. (2020) Low-code autoML-augmented data pipeline – A review and experiments. Journal of Physics: Conference Series, 1828 012015. https://doi.org/10.1088/1742-6596/1828/1/012015 .
12. Sarangpure, N., Dhamde, V., Roge, A., Doye, J., Patle, S., Tamboli, S. (2023). Automating the machine learning process using PyCaret and streamlit. 2023 2nd International Conference for Innovation in Technology (INOCON). https://doi.org/10.1109/INOCON57975.2023.10101357.
13. Davis, J. Goadrich, M. (2006). The relationship between precision-recall and ROC curves. in Proceedings of the 23rd International Conference on Machine learning, 2006.