Credit Card Fraud Detection

Group 12: Wang Xinyi, Liu Qin, Li Hanlong, Li Yao, Miao Jiayi

Table of Contents

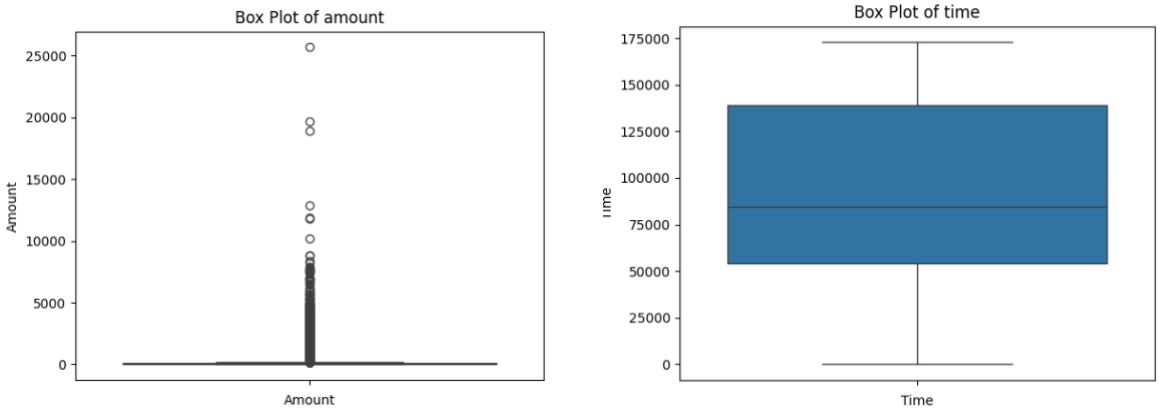
Introduction

The dataset featured in this report is derived from the widely recognized Kaggle resource, known as the "Credit Card Fraud Detection" dataset [1]. It consists of transactions made by European cardholders in September 2013, where a mere 492 transactions out of 284,807 are fraudulent, making it highly imbalanced and thus simulating real-world data challenges.

The primary objective of this analysis is to construct a predictive model capable of classifying and identifying fraudulent transactions with high accuracy. The challenge lies in the imbalance nature of the dataset, and we will discuss the resampling methods in later sections. Then, we perform several machine learning models such as KNN, Logistic Regression, XGBoost, Random Forest and SVM to find the best performing model that can capture the fraud activities in the dataset.

Data Preprocessing

We have 28 standardized principal components that do not require special pre-processing. The remaining two features time and AMOUNT. By observing the box plot shown below, the distribution of their values was not uniform, especially the feature AMOUNT. Under this circumstance, we used robust scalar to do standardization: use the median instead of mean, IQR instead of the standard deviation. This method did standardization and maintained the shape of the dataset's distribution.



Robust Scaler

RobustScaler is a data scaling technique that uses the median and interquartile range (IQR) to adjust the data. This method is particularly useful when dealing with data that contains extreme values or outliers. What sets RobustScaler apart is its ability to maintain the shape and characteristics of the original data distribution even in the presence of these extreme values. RobustScaler is resilient to outliers.

Data Resampling

Various data resampling methods are used (including SMOTE oversample, randomized undersample, ADASYN oversample). ADASYN oversampling methos, which is based on

smote oversample and uses KNN to compute the density of default train point to determine how sparse an observation is. This gives more weight to the "rare" default observations, so that our model can learn different default cases better.

$$densities = 1/(distances.sum(axis = 1)/5)$$

Stratified Cross-validation

In the context of an imbalanced dataset, it's critical to ensure that both the training and testing sets have a fair representation of positive and negative samples. To achieve this balance, a stratified splitting method is employed. This method guarantees that, in each subset or fold of the data, the relative proportions of the different target categories closely resemble the overall distribution in the entire dataset. In this cross-validation method,

Methods

Logistic Regression with Lasso

Logistic regression is a classification algorithm employed for predicting the probabilities of an instance belonging to a specific class in binary scenarios. It models the relationship between target variable and features using sigmoid function, restricting the output to 0 and 1. In addition, lasso is adding L1-regularization like kicking out irrelevant features by setting their coefficients to zero.

In training the dataset, we set hyperparameters as follows. The solver is set to 'sag', and the maximum number of iterations is set to 100,000. The regularization parameter (C) is set to 10. We used random search to tune the optimal parameter. Then, we fitted the model with test dataset and made predictions. The confusion matrix we got is shown below.

```
logistic regression
       0     1
0  51919  4945
1      6    92
              precision    recall  f1-score   support

           0       1.00      0.91      0.95     56864
           1       0.02      0.94      0.04        98

    accuracy                           0.91     56962
   macro avg       0.51      0.93      0.50     56962
weighted avg       1.00      0.91      0.95     56962
```

Lasso Regression introduced a penalty term that could precisely drive some coefficients to zero. This inherent feature selection property is particularly advantageous when dealing with many features based on the dataset. By effectively disregarding irrelevant features, Lasso could enhance the model’s simplicity, reduce the risk of overfitting, and thus achieved better generalization performance on the dataset. However, although the accuracy is high, the precision is very low.

KNN

KNN algorithm functions by classifying data points based on the majority class among their k nearest neighbors. The pivotal hyperparameter in KNN is denoted as K, representing the number of neighbors considered for classification.

To ascertain the optimal value for K which corresponds to minimum mean error, a robust cross-validation procedure with grid research was applied to the under-sampling dataset. The objective was to identify the K value that yields the minimum mean error. Through tuning the range of 1 to 30 with 1 step size, the optimal K was 5. Then, the model was trained on the ADASYN-sampling dataset and its performance was assessed on a fixed test dataset. The KNN demonstrates notable precision in forecasting Class 0 and Class 1.

KNN

	0	1
0	56769	95
1	12	86

	precision	recall	f1-score	support
0	1.00	1.00	1.00	56864
1	0.48	0.88	0.62	98
accuracy			1.00	56962
macro avg	0.74	0.94	0.81	56962
weighted avg	1.00	1.00	1.00	56962

Random Forest

Random Forest, a tree-based ensemble method, combines the "Bagging" technique by training multiple decision trees on bootstrapped data with "Random Selection" of a subset of features at each split.

For this model, we first aimed to optimize the model's hyperparameters, specifically, the depth of trees selected ('max_depth') and the number of decision trees ('n_estimators'). Cross-validation with hyper-opt research process was undertaken to identify the optimal hyperparameter combination of 'max_depth' from 3 to 10 at step size 1 and ('n_estimators') from 50 to 200 at step size 1. After tuning, the optimal 'max_depth' is 9 and 'n_estimators' is 150.

We applied the Random Forest algorithm with tuned parameters to construct a model on the ADASYN-sampling dataset. This model was then utilized to make predictions on the test dataset.

The resulting confusion matrix facilitated the computation of crucial performance metrics, including Accuracy, Recall, Precision, F1 Score. Furthermore, an assessment of the model's quality was accomplished by calculating the PR curve in the comparison of other models. It is noteworthy that Random Forest exhibited excellent predictive performance in accurately predicting instances labeled as 1 and 0.

```
random forest
       0    1
0  56732  132
1     11   87
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     56864
           1       0.40      0.89      0.55        98

    accuracy                           1.00     56962
   macro avg       0.70      0.94      0.77     56962
weighted avg       1.00      1.00      1.00     56962
```

SVM

We applied SVM, SVM with bagging and kernel SVM based on original data, transforming the data from the original feature space to a higher-dimensional space allowing for effective class discrimination. However, the results were similar with no significant difference. Considering computational complexity, we chose linear kernel, the simplest model, to do the further tuning hyperparameter.

In the case of linear kernel, cross-validation processd with randomized search was undertaken in the under-sampling dataset to identify the optimal hyperparameter C, the regularization parameter that controls the trade-off between having a smooth decision boundary and classifying the training points correctly. Over the range (0.001, 0.01, 0.1, 1, 10, 100, 1000), the optimal C was determined to be 1. Subsequently, the model was trained on the ADASYN-sampling using C=1, and its performance was assessed on a fixed test dataset.

```
SVM
       0     1
0  54654  2210
1      6    92
              precision    recall  f1-score   support

           0       1.00      0.96      0.98     56864
           1       0.04      0.94      0.08        98

    accuracy                           0.96     56962
   macro avg       0.52      0.95      0.53     56962
weighted avg       1.00      0.96      0.98     56962
```

While demonstrating commendable accuracy in predicting instances labeled as 1, faces discernible challenges in effectively recognizing and learning the intricate features associated with class labeled as 0 compared with other models. This difficulty in accurately discerning instances marked by a claim label of 0 suggested a limitation in the model's capacity to discern and generalize from a subset of the data characterized by this specific classification.

XGBoost

XGBoost is an algorithm that employs decision trees iteratively to improve predictions by addressing errors from preceding models.

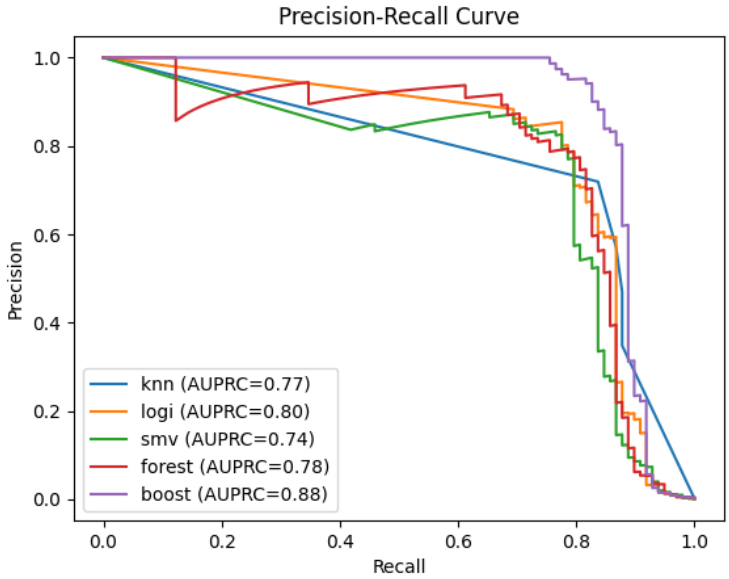
In the pursuit of optimizing XGBoost performance, the tuning of key parameters, including learning rate, max_depth (The interaction depth), and n estimator (number of decision trees) is crucial. Use hyper-opt (which is a type of automated research) to iterate through hyperparameter grid to determine the optimal value. The range of interaction depth is set from 3 to 10 with step size 1, n estimator is from 50 to 200 with step size 1, and learning rate is set from log(0.01) to log(0.03).
The XGBoost algorithm was employed on the training dataset with tuned parameters to build a model. Subsequently, the model was applied to predict outcomes on the test dataset. The performance is shown below:

```
xgboost
       0   1
0  56841  23
1     12  86
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     56864
           1       0.79      0.88      0.83        98

    accuracy                           1.00     56962
   macro avg       0.89      0.94      0.92     56962
weighted avg       1.00      1.00      1.00     56962
```

The precision, recall, f1-score and accuracy were very high. Suggesting that the XGBoost model is effective in capturing patterns and making accurate predictions on the given data.

Results Comparison and Analysis

	Confusion Matrix	Overall Accuracy	Precision	Recall	F1 score
KNN	56767 97 12 86	0.9981	0.9983	0.9998	0.999
Logistic Regression	51919 4945 6 92	0.9131	0.913	0.9999	0.9545
Random Forest	56000 864 10 88	0.9847	0.9848	0.9998	0.9923
SVM	54654 2210 6 92	0.9611	0.9611	0.9999	0.9801
XG boost	56841 23 12 86	0.9994	0.9996	0.9998	0.9997



We have the results of each model in the first picture, and the AUPCR (Area under the PR curve) in the second picture. KNN exhibits excellent precision and recall, leading to an almost perfect F1 score. Despite this, the AUPRC is the second lowest among the models, suggesting that while KNN is highly accurate, it may not perform as well across different thresholds when considering the balance between precision and recall. However, even Logistic regression and Random Forest have lower F1 scores, accuracy, and precisions compared to KNN, their AUPRC are higher than KNN's, indicating they may offer a better balance between precision and recall across different thresholds. SVM and Logistic Regression appear to have a significant number of false positives as indicated in the confusion matrix, which explains their lower precisions. XGBoost presents the best balance of precision and recall, with F1 score comparable to KNN. It also has the highest AUPRC at 0.88, indicating a superior balance between precision and recall across various thresholds and making it the most consistent model in terms of performance.
Based on these results, XGBoost is the best model among those tested. It did not only provide high precision and recall, resulting in a high F1 score, but also has the highest AUPRC, indicating its performance.

Conclusion

In this project, we use and compare the different resampling methods including random-under sampling, SMOTE over sampling, and ADASYN sampling. Then we utilize several cross-validation methods such as hyperopt, random search and grid search to split the data. Next, we select appropriate evaluation metrics (recall rate of default, precision of default, accuracy, Precision-Recall curve) because we are more interested in predicting default. Several robust and accurate fraud detection models that perform well in unbalanced/balanced datasets including logistic regression, XGBoost, SVM, and random Forest are developed and ready to compare. Hyperparameter tuning is essential for achieving optimal models. Subsequently, we execute an experimental study, analyze

results with appropriate evaluation metrics, and evaluate multiple models, aiming to find the best-performing model customized for credit card fraud detection: (Random Forest and XG boost). Despite encountering certain challenges during our research, we systematically addressed these issues through extensive and rigorous investigation.

Future Improvement

There remains potential for further enhancement. First, we can use deep learning models such as Feedforward Neural Networks for credit card fraud detection. Second, we can incorporate Variational Auto encoders (VAE) and Generate Adversarial Networks (GAN) to generate examples of minority classes for unbalanced datasets and do data denoising. Third, we can use Kappa to analyze the confusion matrix in unbalanced datasets. Fourth, we can use ROC curves as evaluation metrics. Lastly, we intend to consider changing the classification threshold using "predict_proba()" to control precision and recall.

Reference

[1] https://www.kaggle.com/mlg-ulb/creditcardfraud