Fradulent Credit Card Transaction Detection

Min Luo
Department of Statistics and Data Science
University of Central Florida
Orlando, FL
min.luo@knights.ucf.edu

***Abstract*—Credit card transaction has been the primary form of payment for many consumers. Being able to detect and stop a fraudulent transaction is important to the cardholder's security and the credit card company's accountability. In fitting a fraud detection model, the nature of imbalance in the number of fraudulent versus normal transactions in the training data needs to be dealt with carefully in pre-processing the data. In this project, xgboost models were used to predict whether a credit card transaction is fraudulent using a dataset containing transaction information. Re-balancing methods like random under-sampling, SMOTE, and cost-sensitive learning were used to balance the number of observations in each class. A grid search cross validation approach was used to determine the optimal values for the hyperparameters of interest.**

Keywords—credit card transaction, fraud detection, fraudulent, classification, XGBoost, undersamping, SMOTE, imbalance data

I. INTRODUCTION

Online and in-person credit card transactions occur every day, and due to the rapid growth of e-commerce and financial technology, consumers are more and more likely to use their credit card rather than cash. It's critical for credit card companies to recognize fraudulent credit card transactions to ensure the card holder's security and the credit card company's accountability. In addition, fraudulent credit card transactions could also indicate potential identity theft. Alerting the individuals early when suspicious activities were identified may prevent further malicious activities.

Among the transactions made, the proportion of fraudulent transactions is often very small. When building a classification model, attention should be paid to identify the fraudulent cases over maximizing the accuracy. Since the accuracy of a classifier is maximized if the model predicts all observations to be in the majority class when the dataset is unbalanced.

In this project, different pre-processing methods were used, and an XGBoost model was fitted to predict whether a credit card transaction is fraudulent in an unbalanced dataset. The effects of over- and under-sampling methods, and cost-sensitive learning were also analyzed.

II. DATA

A. Data Source

The dataset used in this project was originally collected and analyzed in a collaborative research of Worldline and the Machine Learning Group of ULB (Université Libre de Bruxelles). It's currently hosted on Kaggle [1].

B. Data Description

The dataset contains two days of credit card transactions made in September 2013 by European cardholders. There are 284,807 observations, and 30 predictor features, where each observation is a transaction made. Of the 30 predictors, 28 of them, "V1" to "V28", are the result of a principal component analysis (PCA) transformation. This was done to protect the cardholders' confidentiality, and no additional information was provided regarding to those variables. This limits the ability to perform feature engineering using domain knowledge.

Variables "time" and "amount" were not transformed. The "time" variable is the number of seconds elapsed since the first transaction occurred in the dataset. However, the time when the first transaction took place during the day is unknown. By looking at the range of time, we can verify that the dataset covers a 48-hour period. For transactions that occurred at the same time, they have the same time value.

Table 1 shows a summary of the variables in this dataset. All the predictor variables are numerical, and there is no missing value in the dataset.

TABLE I. SUMMARY OF VARIABLES IN DATASET

Variable	Type	Description
Class	Categorical	Target variable. 1 = fraudualent transaction, 0 = non-fraudulent transaction
V1 to V28	Numerical	Credit card transaction information resulted from a PCA transformation
Time	Numerical	Number of seconds elapsed between this transaction and the first transaction in the dataset
Amount	Numerical	Transaction amount

III. DATA EXPLORATION ANALYSIS

The dataset is highly unbalanced; the fraudulent class only accounts for 0.1727% of the dataset, i.e. there are only 492 frauds out of 284,807 transactions. To investigate if the fraudulent transactions behaved differently, the distribution of the predictor variables were analyzed.

Figure 1 shows the boxplot of the transaction amount. The distribution of amount is highly skewed to the right. To have a clearer visualization the distribution, the outliers were removed. The outliers here are any values that are outside of 3 times the inter-quantile range (3IQR). The average transaction amount in the original dataset is 88.350, with a standard deviation of

250.120. Next, to see if there is a difference between the normal and fraudulent transactions, boxplots of amount by class was plotted as shown in figure 2.

There is not a significant difference between the transaction amount in general between the two classes when outliers were removed. In the original dataset, the normal transaction amounts have a lot more outliers compared to the fraudulent cases. For anomaly detection, we may consider removing the extreme outliers, however, the trade-off for removing the outliers is the loss of information.

There is not a significant difference between the transaction amount in general between the two classes when outliers were removed. In the original dataset, the normal transaction amounts have a lot more outliers compared to the fraudulent cases.

Though the starting time in the dataset is unknown, we can visualize the time to see if the fraudulent and normal transactions occur during certain time frames. Figure 3 shows the density plot of the time variable. The bi-modal shape of the distribution is due to the two-day period in the dataset. Scatter plots of the transaction amount over time were plotted as shown in figure 4. In each 24-hour period, the number of transactions is the highest in the middle. However, the time of transaction does not appear to matter for the type of transactions.

The heatmap of the all the variables is shown in figure 5 (top). The variables V1 to V28 are not correlated as expected, since they resulted from a PCA transformation. There is moderate linear correlation between the variables. For example, the correlation coefficient for V2 and amount is -0.53, and it's V3 and time has a correlation coefficient of -0.42. However, the presence of the outliers may have skewed the correlation between the variables.

The imbalance issue with the dataset will cause classier models that favor accuracy to predict mostly observation as non-fraudulent. Hence, accuracy should not be used as the sole model evaluation method here.

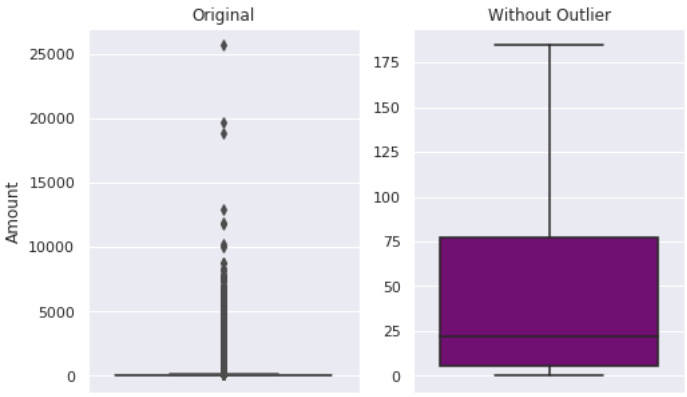

Fig. 1. Boxplot of transaction amount with the original dataset (left) and with the outliers removed (right).

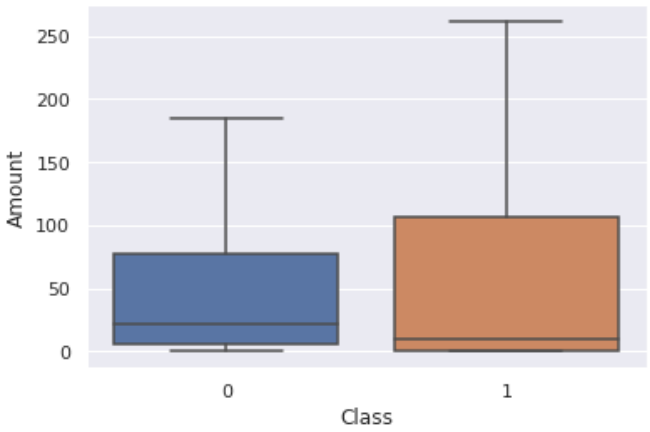

Fig. 2. Boxplot of the transaction amount by Class (0 = normal, 1 = fraud) with outliers removed.

Fig. 3. Density plot of the varaible time of all observations (top), normal transactions (middle), and fradulent transactions (bottom).

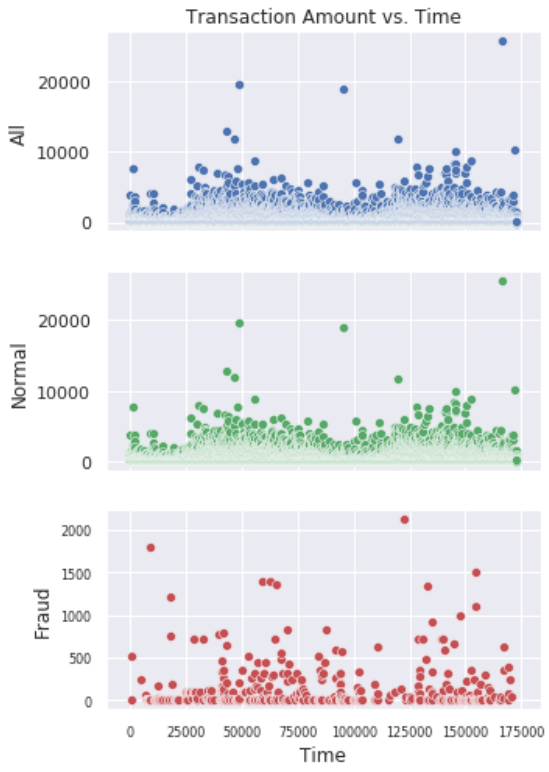

Fig. 4. Scatter plot of transaction amount over time of all observations (top), normal transactions (middle), and fradulent transactions (bottom).

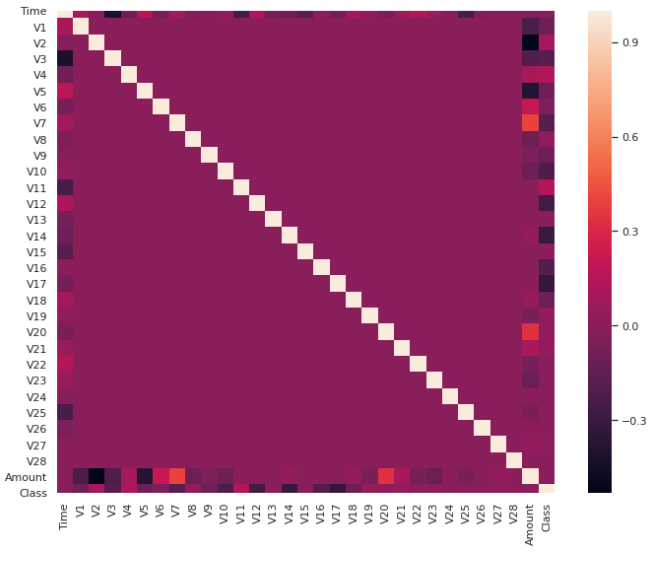

Fig. 5. Heatmap of all variables in the dataset.

IV. MODEL SELECTION

Prior to fitting the model, some pre-processing steps were taken to overcome the aforementioned issues within the dataset, as discussed below. In addition, since the 28 PCA features were standardized, standardization was also applied to the untransformed features.

A. *Handling Unbalanced Data*

There are two common approaches in handling unbalanced data: rebalancing in pre-processing and using a cost-sensitive learning method.

The motivation for doing pre-processing to balance an unbalanced dataset is to minimize the biased outputs from the classifier. The balancing is often done via over- or under-sampling. In this project, random under-sampling (RUS) and synthetic minority over-sampling technique (SMOTE) were used. RUS is the simplest case of under-sampling, where a simple random sample of the size of the minority class was generated from the majority class. However, under-sampling risks loss of potential useful information. And by discarding the observations in the majority class, the resulted sample may not be a good representation of the population distribution.

SMOTE is an over-sampling method where new observations of the minority class are synthesized based on the existing minority observations and parameters defined. Over-sampling methods mitigates the loss of information compared to under-sampling. However, in SMOTE, the synthetic samples do not account for neighboring observations from other classes, hence may result in overlapping of classes, and consequently introduce additional noise.

In cost-sensitive learning, the cost of misclassification is built directly into the algorithm. While classification models traditionally assume all misclassification errors have the same costs, cost-sensitive learning assigns different costs by class. One way to implement a cost-sensitive learning model is to give a different weight or cost to each class. By assigning a higher cost to misclassifying a fraudulent transaction as normal, the model's bias is hence reduced.

In addition, as suggested in [1], since confusion matrix accuracy is not meaningful given the imbalance ratio, using the Area Under the Precision-Recall Curve (AUPRC) to measure accuracy is recommended.

B. *Standardization*

Traditional standardization method of subtracting the mean and dividing by the standard deviation may be affected by the existing outliers in the dataset. Hence, the effect of using a robust standardization method was explored. In the robust standardization, the median is subtracted from the sample then it is divided by the IQR.

C. *XGBoost*

XGBoost (eXtreme Gradient Boosting) is a robust and effective machine learning technique. It's an advanced implementation of the gradient boosting machine (GBM) algorithm. GBM is a ensemble learning method based on decision tree where each subsequent tree focuses on the error from the previous tree. Due to its flexibility, the model is not bounded to the structure of the data. To prevent overfitting for a flexible model like GBM, hyperparameters like tree depth, learning rate, and number of nodes can be tuned.

The advantages of xgboost over GBM can be categorized into three different areas as outlined in [6]: model features, system features, and algorithm features. Here, we will discuss the some of the main advantages in the model features.

To determine the optimal values for the hyperparameters, grid search or previous knowledge is often used. Without the xgboost approach in implementing gradient boosting, GBM could take up a lot of resources, especially with a large dataset and hyperparameter grid. While xgboost still follows the gradient boosting principle, it uses a regularized model to control overfitting, hence improving the model's ability to generalize. To increase training speed, xgboost adapts a stochastic gradient boosting method, where it sub-samples at the row, column, and column per split levels.

Popular machine learning libraries like scikit-learn in python and h2o and caret in R can be used to implement xgboost. These libraries provide a system that supports features like parallelization and out-of-core computing to further enhance the speed of xgboost. This project primarily used the xgboost library with scikit-learn in python.

D. Model Metrics

Each model was evaluated by its precision, recall, and AUC (area under receiver operating characteristic curve). Here, precision, or true positive rate, refers to the proportion of true fraudulent transactions identified by the model; and recall is the proportion of actual fraudulent transactions out of the transactions classified as fraud by the model. Due to the imbalance in the dataset, controlling the recall and precision will prevent the model from favoring the majority class. From an application standpoint, the cost of missing a fraudulent transaction is higher than falsely identifying a normal transaction as fraudulent. The AUC value represents the model's ability to discriminate between the two classes, a higher AUC value above 0.5 indicates a higher discriminatory ability. Hence, an optimal model should have a recall, precision, and AUC.

E. Model Selection

Due to the large sample size and the amount of hyperparameters to be tuned, the model selection process was broken down into two different parts: prototyping and fine-tuning.

First, in prototyping, the project explored five different pre-processing pipelines as shown in table 2. The effect of removing outliers, different standardization methods and re-balancing methods were investigated. An xgboost model was fitted after each pre-processing pipeline. A 5-fold cross validation process was done for the 5 models and their cross-validation and training metrics were compared. The best models were then selected into the fine-tuning process.

Secondly, in the fine-tuning part, the hyperparameters of the models selected previously were determined over a range of values by using a grid search 5-fold cross validation process. In grid search, a range of values were pre-defined for each hyperparameter to be tuned, hence forming a grid. For a specific combination of values of hyperparameters, a 5-fold cross validation process was implemented as before.

The rationale for not performing grid search in the first part with the 5 initial models is that grid search is computationally expensive in general. Given a large dataset, a grid search could take a long training time, even more so with cross validation. With the dataset used in this project with over 200,000 observations, the training time of each fold in cross validation was at least 40 to 70 seconds on average, which translates to at least 200 to 350 seconds (3.33 to 5.83 minutes) for a 5-fold cross validation process. However, with grid search, this training time would be substantially longer.

Consider a grid with 5 hyperparameters, each with 3 values of interest. In a grid search, 3^5 models need to be trained in order to consider all the possible combinations. With one 5-fold cross validation taking approximately 3 minutes to complete, to implement a grid search 5-fold cross validation with such a grid would take at least 13 hours for each pipeline. This is highly inefficient.

TABLE II. MODEL PRE-PROCESSING PIPELINES. THE COLUMNS FROM LEFT TO RIGHT REPRESENTS THE PRE-PROCESSING STEPS APPLIED IN THAT ORDER. A GREY BOX INDICATES THAT THE STEP WAS NOT PERFORMED.

	Standardization	Re-balancing
Model 1	Robust	Weighted Classes
Model 2	Robust	SMOTE
Model 3	Mean/Stand. Dev	SMOTE
Model 4	Robust	RUS
Model 5	Robust	

To reduce the training time in a grid search process, methods such as stochastic gradient boosting or a sequential grid search approach could be used.

V. RESULT AND DISCUSSION

The result of the prototyping step with the initial 5 models is shown in table 3. The scores shown are the average across the 5-fold cross validation. Models 1 and 5 have the best performance overall. Models 2, 3, and, 4 all involved under-sampling or over-sampling in their pre-processing step. Their recall scores are substantially lower than that of models 1 and 5.

In addition, since models 2 and 3 involved over-sampling, i.e. the size of the training set increased, the training time was also higher for those models. In contrast, model 4 used under-sampling, its training time was much lower compared to the others.

TABLE III. SUMMARY OF RESULTS OF THE INITIAL 5 MODELS.

	AUC		Precision		Recall	
	Train	Test	Train	Test	Train	Test
Model 1	0.9974	0.9807	0.8360	0.7564	0.9688	0.8903
Model 2	0.9986	0.9768	0.9698	0.8820	0.1349	0.0925
Model 3	0.9985	0.9758	0.9642	0.8712	0.1383	0.1052
Model 4	0.9984	0.9789	0.9965	0.9029	0.0564	0.0502
Model 5	0.9974	0.9807	0.8360	0.7564	0.9688	0.8903

Model 2 with robust standardization and SMOTE were moved into the fine-tuning step along with models 1 and 5 to see if using different values for the hyperparameters could improve its recall scores.

The result of the fine-tuning step is shown in table 4. Similarly, these scores are the average across the cross validation in each model. The three model all performed well in terms of AUC, all of them have an AUC above 0.95 and the difference between the training and test AUC was small. In terms of sensitivity, model 2 with SMOTE outperformed the other 2 models. However, when looking at the precision, the proportion for model 2 is substantially lower than the other 2 models, indicating that SOMTE was not able to solve the imbalance issues here, and the model prediction was bias toward the majority class

TABLE IV. SUMMARY OF RESULT OF GRID SEARCH. TR. REFERS TO THE TRAINING SET AND TE. REFERS TO THE TEST SET

		Model 1 (Wt. Class)	Model 2 (SMOTE)	Model 5 (No Rebalance)
AUC	Tr.	0.9989	0.9988	0.9863
	Te.	0.9711	0.9726	0.9644
Precision	Tr.	0.9470	0.9785	0.8343
	Te.	0.8317	0.8730	0.7758
Recall	Tr.	0.8599	0.3058	0.9788
	Te.	0.6445	0.1967	0.8340

Overall, model 5 without any pre-processing procedures to rebalance the dataset had the best performance.

Note that in the grid search step, only 2 hyper parameters were tuned based on the impact on the model: max depth (max_depth) and minimum child weight (min_child_weight). Max depth is the maximum number of nodes in a tree. The larger the depth, the more complex the model is. And xgboost consumes the memory excessively as the depth increases. Minimum child weight is the minimum sum of instance weight needed in order to form a child node. The larger the minimum child weight is, the more conservative the algorithm is. In addition, for the weighted class or cost sensitivity learning model (model 1), the balance between the weights of the majority and minority classes (scale_pos_weight) was also tuned. A larger value for this balance gives a higher weight for the minority class.

Due to the computational intensity from the algorithm and the size of the data set, only a small range of hypermeters and values were explored. A larger hyperparameter grid could potentially further improve the precision and recall of the model.

VI. CONCLUSION

Imbalance data is a typical characteristic in dataset used for fitting fraud detection models. This project explored 5 different approaches to handle the data imbalance issue using a xgboost model. Through grid search cross validation, optimal values for a subset of hyperparameters were determined.

For a xgboost model, the pre-processing procedures used to rebalance the classes lowered the overall performance of the model. Since xgboost is a higher flexible model, a thoughtfully tuned model is likely to have better predictive ability than using the re-balancing methods during pre-processing.

The fraud detection problem was treated as a binary classification problem here. While, it's possible to do so, an anomaly detection method may be more appropriate. In anomaly detection, the behaviors of the abnormal observations could differ significantly from each other, while in a binary classification problem, it's assumed that the observations in the normal class and the abnormal class behave similarly within their own class. Hence, alternatively, anomaly detection methods such as one-class SVM could be used.

REFERENCES

[1] Machine Learning Group. "Credit Card Fraud Detection." Kaggle, 23 Mar. 2018, www.kaggle.com/mlg-ulb/creditcardfraud

[2] Mishra, Satwik. *Handling Imbalanced Data: SMOTE vs. Random Undersampling*. International Research Journal of Engineering and Technology, Aug. 2017, https://www.irjet.net/archives/V4/i8/IRJET-V4I857.pdf.

[3] Altini, Marco. *Dealing with Imbalanced Data: Undersampling, Oversampling and Proper Cross-Validation*. 17 Aug. 2015, https://www.marcoaltini.com/blog/dealing-with-imbalanced-data-undersampling-oversampling-and-proper-cross-validation.

[4] Credit Card Fraud Statistics. (2019, October). Retrieved November 2019, from https://shiftprocessing.com/credit-card-fraud-statistics/.

[5] Holmes, Tamara E. "Credit Card Market Share Statistics." CreditCards.com, 12 Sept. 2019, www.creditcards.com/credit-card-news/market-share-statistics.php.

[6] Brownlee, Jason. "A Gentle Introduction to XGBoost for Applied Machine Learning." Machine Learning Mastery, 21 Aug. 2019, https://machinelearningmastery.com/gentle-introduction-xgboost-applied-machine-learning/.

[7] "Introduction to Boosted Trees." Introduction to Boosted Trees - Xgboost 1.0.0-SNAPSHOT Documentation, https://xgboost.readthedocs.io/en/latest/tutorials/model.html.

[8] Analytics Vidhya. (2019). How to handle Imbalanced Classification Problems in machine learning?. [online] Available at: https://www.analyticsvidhya.com/blog/2017/03/imbalanced-classification-problem/ [Accessed 24 Nov. 2019].