Credit Card Fraud Detection

Zhihao Chen

May 6, 2023

Abstract

This project will analyze a highly imbalanced dataset, resample the data using 3 different methods, train through 3 different models, and compare them using 3 different metrics. I will finally give a pipeline of how we can deal with such kind of problems in the future.

This is my last project in the Data Science Master Program at Penn, and also my last project as a student before joining a bank as a data scientist. This is why I choose this credit card fraud detection topic, and I will not only use the knowledge taught in this course, but also what I've learned so far to complete this project.

1 Introduction

1.1 Overview

For bank companies, credit cards are an important product, and credit card fraud is a significant threat to retain high profitable customers. In this project, my goal is to try some different re-sampling methods, and build some machine learning models to predict those frauds which involves an unauthorized taking of other's credit card information for the purpose of charging purchases to the account.

The dataset I used contains 284,807 transactions made by credit cards in 2 days in September 2013 by European cardholders. Due to confidentiality issues, our dependent variables are 28 principal components obtained from PCA, along with 'time' and 'amount'.

Setup:

Given $\mathcal{D} = \{Y_i, X_{1i}, X_{2i}, ..., X_{29i}\}$, where

- $i = 1, 2, ..., 284807$
- $Y_i \in \{0, 1\}$
- $X_{1i}, ..., X_{28i} \in \mathbf{R}$ are the principal components
- $X_{29i} \in [0, \infty)$ is the amount of transactions

Goal:

Find a model $f : X \to Y$ where we want to maxmimize the recall and the area under the ROC curve.

1.2 Challenges

The dataset is highly imbalanced: within the total of 284,807 transactions, only 492 (around 0.17%) transactions are considered fraud. This could be a big problem for a machine learning task. In this project, I will try both undersampling and oversampling methods to balance our data, and see how they will help in improving the model performance.

1.3 Current State-of-the-art and Potential Limitations

In the current industry, one of the most widely used techniques on this problem is still logistic regression, not only because it is easy to understand, but also due to the regulation on the model and data usage. Other commonly used models are tree ensembles methods like XGBoost, which usually has a better performance. However, these models have met their limits on the performance of problems like fraud detection these years. Therefore, in this project I will try some multi-layer neural networks models, and compare them to the traditional models discussed above.

2 Method

2.1 Data Preparation

In our dataset, 492 out of the 284,807 data is fraud, and the rest 284,315 is non-fraud.

Let's take a look at the distribution of 28 principle components for fraud and non-fraud data:

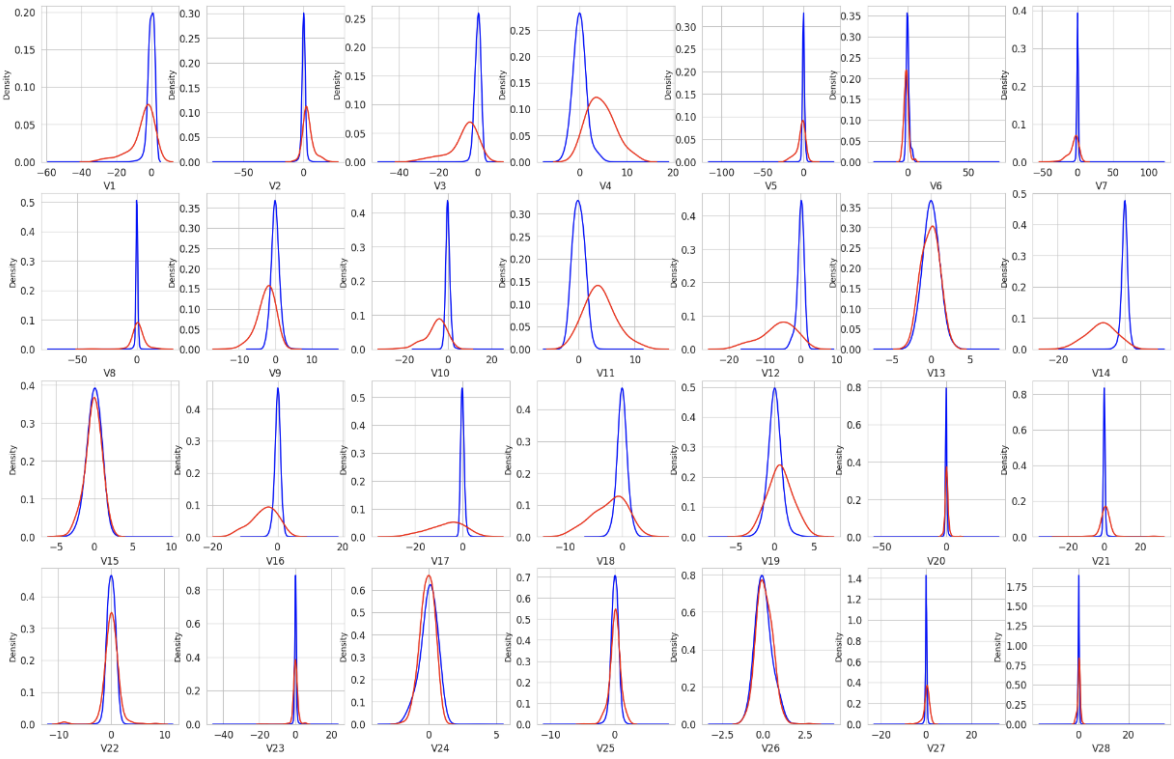

Figure 1: Distribution of features (Red is fraud and Blue is non-frauds).

We can see that there are some obvious differences in the distribution, and thus in the next part, we will sample from the fraud data distribution to balance our dataset.

Then we split the data into 70% train data and 30% test data using sklearn, and train the machine learning models on the train data, while check their performance on the test data.

The first 28 features are the principal components, so I will just keep them. And for the 'Amount' variable, which varies from \$0 to over \$20,000, I used StandardScaler() to standardize the train 'Amount' variable, and also transform the test 'Amount' variable.

2.2 Re-sampling Methods

One big problem for the fraud detection problem is the highly imbalanced data as we saw in this credit card fraud dataset. Feeding imbalanced data to the classifiers can make them biased in favor of the majority class and ignore the minority. Therefore, in this section, I will introduce some resampling methods that I will use in order to solve this problem.

2.2.1 Under-sampling: Cluster Centroids

This method undersamples the majority class by replacing a cluster of majority samples This method finds the clusters of majority class with K-mean algorithms. Then it keeps the cluster centroids of the N clusters as the new majority samples.
The main advantage of Cluster Centroids is that it preserves the information contained in the majority class samples, while reducing their number to balance the dataset. This can help to avoid overfitting and improve the generalization performance of the classifier. Additionally, it can be faster and less memory-intensive.
However, it can result in the loss of information if the clustering algorithm fails to identify all the important patterns in the majority class samples, or the majority class samples are not clustered and spread out evenly across the feature space, then this technique may not be effective.

2.2.2 Random Over-sampling

Random Over-sampling allows us to oversample the minority class by randomly duplicating its samples until the number of samples in both classes is balanced.
The main advantage of random oversampling is that it is a simple and easy-to-implement technique that can improve the performance of classifiers on imbalanced datasets. By increasing the number of minority class samples, it allows the classifier to learn from more diverse examples and better capture the underlying patterns in the data. Additionally, it does not require any prior knowledge or assumptions about the data distribution, and can be applied to any binary classification problem.
However, it can have some drawbacks such as overfitting due to the oversampled data, especially when the minority class samples are highly clustered or have high variance, then the oversampled data may not accurately reflect the true distribution of the minority class.

2.2.3 SMOTE

SMOTE stands for Synthetic Minority Over-Sampling Technique. It keeps joining the points of the minority class with line segments and then places artificial points on these lines.

The procedures are like:

1. Choose a minority class input vector
2. Find its k nearest neighbors
3. Choose one of these neighbors and place a synthetic point anywhere on the line joining the point under consideration and its chosen neighbor
4. Repeat the steps until data is balanced

SMOTE's advantage over random-oversampling is that the synthetic samples are more representative of the minority class than simply duplicating existing samples. By interpolating between samples, it can create new samples that are located in regions of the feature space that are not covered by existing samples, which can help the classifier to better capture the underlying patterns in the data. However, this could also lead to overfitting, and may not be so accurate when there's no such patterns.

2.3 Machine Learning Models

2.3.1 Logistic Regression

For this binary classification problem, logistic regression is the most commonly used and easy to implement model. I built and trained the model using the binary cross entropy loss in jax, which is faster than using numpy and other pure python functions, and can use GPUs to accelerate computations. This is especially important when dealing with large data and complex models. This data set is already very large, but in real life problem, the dataset is much larger, with billions of transactions every day. So this code is just an example, and I will extend its use in the future.

2.3.2 XGBoost

XGBoost is fitting multiple decision trees, with each tree considering the residual from the previous one, and we combined all the trees at the end. It is a more complex and flexible model than logistic regression. In our problem, XGBoost's ability to capture non-linear relationships and handle missing data can be advantageous in improving the accuracy of predictions. Its robustness to outliers may allow it to better identify fraudulent transactions. Another important advantage for XGBoost is that it's easier to refit if we change our response variable in the future, but if we use logistic regression, there might be a big change in feature selection. Also, XGBoost is good for its fast speed, high accuracy, and controls overfitting.

However, XGBoost has many hyperparameter which is hard to tune, and the final model is complex and hard to explain to the stakeholders. It can also be computationally expensive to train, especially for large datasets. This may be a disadvantage in fraud detection, where real-time detection is often important.

2.3.3 MLP

Multilayer perceptron (MLP) is a type of neural networks that can be used for classification like credit card fraud detection. Compared with logistic regression and XGBoost, MLP may have higher accuracy, but will also be more computationally expensive to train, and even more hyperparameters tuning than XGBoost.

I'm building this MLP model in JAX following what I've learned in this course. I used a weighted binary cross entropy loss in order to penalize more on the false negatives. I generated the data with batch size 128, and I used stochastic gradient descent and adam to optimize the loss function. The layers of my model is an input of 29 units, and output 1 units, with 2 hidden layers of 64 units, while using the ReLU activation function. All the models are trained for 10,000 iterations.

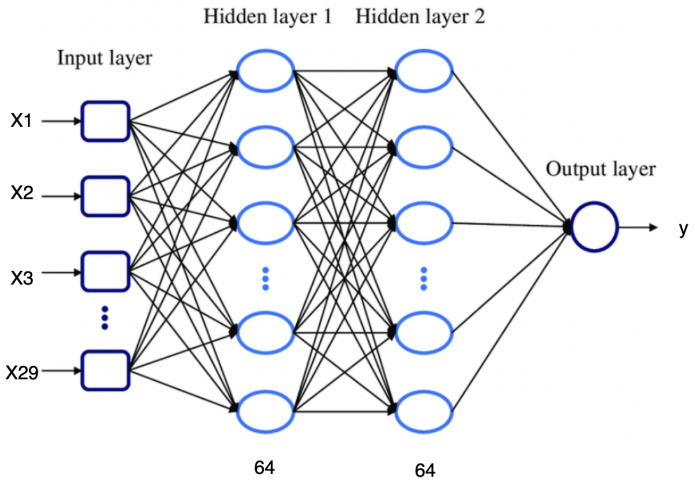

Figure 2: MLP Structure

2.4 Evaluation Metrics

For fraud detection problems, the cost of false negative (**FN**) is much larger than the cost of false positive (**FP**). It is fine when the bank rejects a real transaction, but there will be serious problems when it approves a fraud one. Here I will list some metrics that I use for this project.

2.4.1 Accuracy

Accuracy is the one of the most commonly used evaluation metrics, but it is not that important in this problem, because if our model doesn't do anything and just predict everything to be non-fraud, then the accuracy will be 99.83%. However, it can still help us keep track of and reduce the number of both false positives and false negatives.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

2.4.2 Recall

Recall is the fraction of true positives among all the real positives. As we mentioned earlier, what we really cares about in the fraud detection is real 1's, while we want to minimize the false negatives. Therefore, recall is a perfect evaluation metric in this case.

$$Recall = \frac{TP}{TP + FN} \tag{2}$$

2.4.3 ROC and AUC

A ROC curve plots the true positive rate (TPR) versus the false positive rate (FPR) as a function of the model's threshold for classifying a positive. And the area under the ROC Curve (AUC) measures the entire two-dimensional area underneath the entire ROC curve from (0,0) to (1,1).

$$TPR = \frac{TP}{TP + FN} \qquad FPR = \frac{FP}{FP + TN} \tag{3}$$

3 Results

3.1 Logistic Regression

First, let's look at the performance of each sampling methods on logistic regression model by the following confusion matrices. The original model with the imbalanced data has 18 false negatives and 2,772 false positives.

For the Cluster Centroid under-sampling, the number of false positives has largely decreased, but the false negatives (which is really important as mentioned before) has slightly increased.

On the other hand, for the Random Over-sampling, the number of false negatives has decreased to only 9, but the false positives becomes large (6301, more than 2 times the original).

For the SMOTE, it seems like the best method, since both the false positives and false negatives has decreased, which shows that the re-sampling is actually improving the performance of our original logistic regression model.

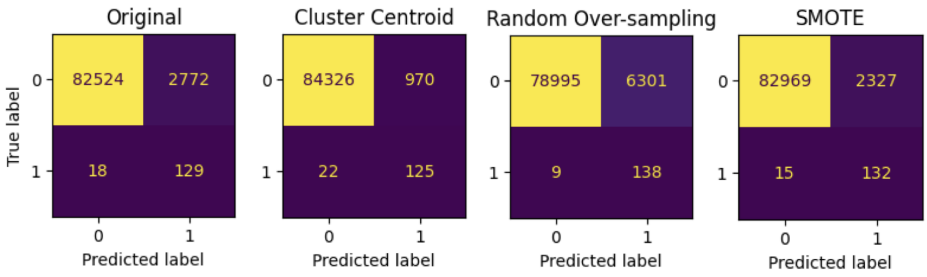

Figure 3: Logistic Regression Confusion Matrix

The below figure is the ROC curves for the 4 models and their AUC. Obiviously, the Cluster Centroid is losing some information in the under-sampling process and get a lowest AUC. While both over-sampling methods have a higher AUC than the original model.

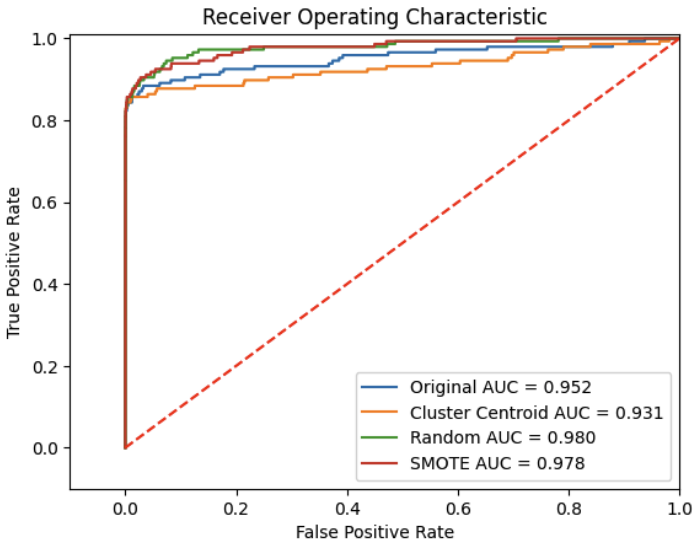

Figure 4: Logistic Regression ROC

Metrics	Original	Cluster Centroid	Random Oversampling	SMOTE
Accuracy	96.7347%	98.8390%	92.6150%	97.2590%
Recall	87.7551%	85.0340%	93.8776%	89.7959%
AUC	95.1968%	93.0927%	97.9608%	97.8451%

Figure 5: Logistic Regression Result

The above table is putting all the evaluation metrics: accuracy, recall, and AUC together for the 4 models, so that it's easier for us to compare them. From the table, we can see that there's a trade-off between accuracy and recall. Undersampling can give us a higher accuracy but lower recall, and over sampling can give us a higher recall but lower accuracy. This is what we need to decide whether we want less false positives, or less false negatives. However, based on all the information above, SMOTE has a higher accuracy, higher recall, and higher AUC. Therefore, if we are going to use logistic regression as our model, I would suggest to use SMOTE to deal with the imbalanced data.

3.2 XGBoost

Then we check the performance of each sampling methods on the XGBoost model. The original model with the imbalanced data has 15 false negatives and 2,039 false positives.

Different from logistic regression, for the Cluster Centroid under-sampling, the number of false positives has increased, and the false negatives has also slightly increased, which shows that this is not a good resampling method to use.

For the Random Over-sampling, the number of false positives has largely decreased, and the false negatives has just increased 2, which seems to be a better method.

For the SMOTE, the number of false negatives is the same as random over-sampling, but the number of false positives is more than 2 times more, so I will still prefer random over-sampling than SMOTE for XGBoost.

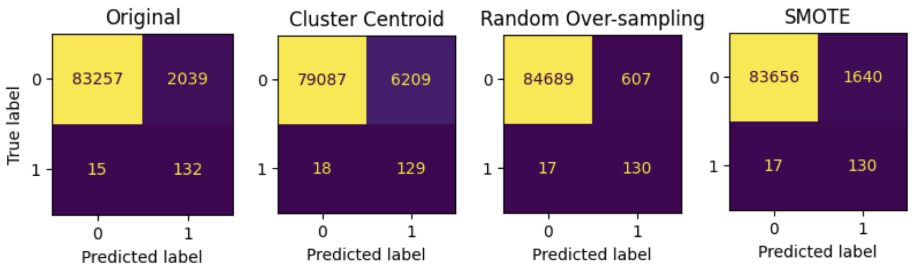

Figure 6: XGBoost Confusion Matrix

From the below ROC curves, we see that the original model already has a high AUC, and Random Over-sampling is very close to it. Cluster Centroid has the lowest AUC, which confirms what we found in the confusion matrix that it has lost some information in the resampling method.

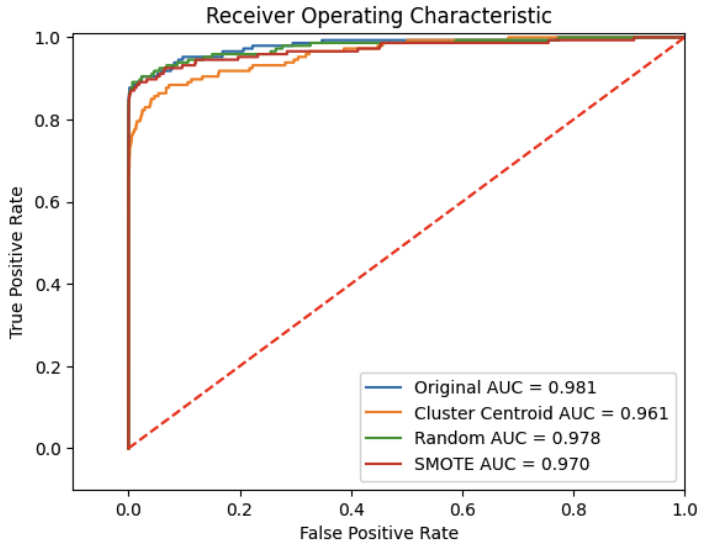

Figure 7: XGBoost ROC

Metrics	Original	Cluster Centroid	Random Oversampling	SMOTE
Accuracy	97.5961%	92.7121%	99.2697%	97.2590%
Recall	89.7959%	87.7551%	88.4354%	89.7959%
AUC	98.0967%	96.0731%	97.7857%	97.0353%

Figure 8: XGBoost Result

After putting all the metrics together, we know that the original model of XGBoost using the imbalanced data already has a very good performance, because XGBoost itself can handle imbalanced data by tuning it properly. If we want to balance the data, I will suggest Random Over-sampling in this case, where it can sacrifice by increase a little bit false negatives, and largely reduce the number of false positives.

3.3 MLP

I trained the MLP using JAX for 10,000 iterations, and the following plots is a showcase of the training process.

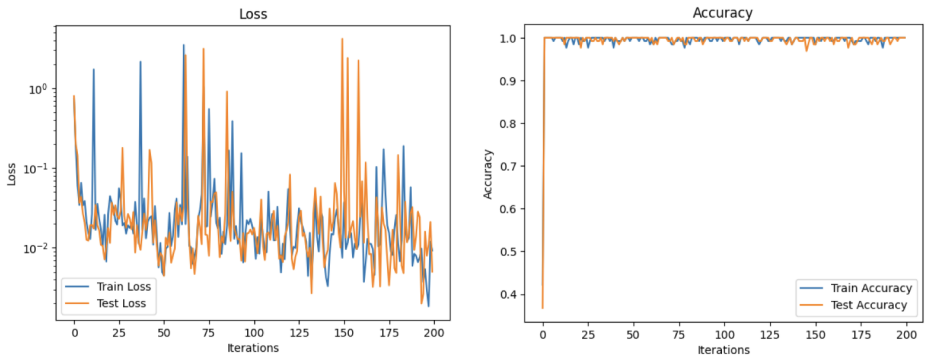

Figure 9: MLP Training

Then let's look at the performance of MLP for each sampling methods. The original model with the imbalanced data has 20 false negatives and 114 false positives, which is already pretty good! Probably because I used the weighted binary cross entropy loss in order to let our model focus more on reducing the false negatives.

The Cluster Centroid under-sampling only has 8 false negatives which is good, but the number of false positives has increased so much to over 10,000 which is not acceptable, and this shows that this undersampling is not a good resampling method for MLP.

The Random Over-sampling and the SMOTE have similar performance, but both the number of false positives and the false negatives is slightly more than the original model. Maybe because these two models need to be tuned again to fit the data, but it seems than for model with high accuracy like MLP, over resampling is adding some nuances to the original data which can lead to overfitting, or adding some useless information there.

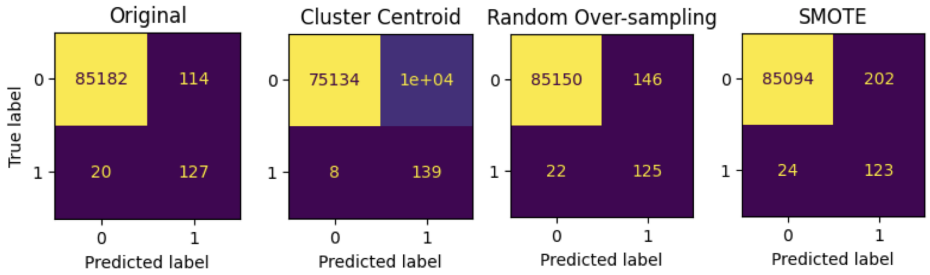

Figure 10: MLP Confusion Matrix

The ROC curve of the Cluster Centroid under-sampling is the best with 0.974 AUC, but as we analyzed earlier, there are too many false positives, which is another thing that we need to consider. And for the rest, the orginal MLP model has a higher AUC than Random Over-sampling and SMOTE.

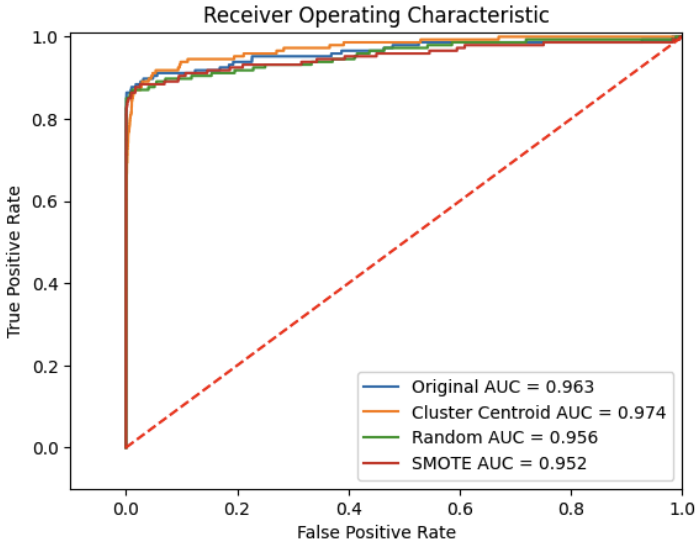

Figure 11: MLP ROC

Metrics	Original	Cluster Centroid	Random Oversampling	SMOTE
Accuracy	99.8432%	88.0973%	99.8034%	99.7355%
Recall	86.3946%	94.5578%	85.0340%	83.6735%
AUC	96.2743%	97.4141%	95.5979%	95.2461%

Figure 12: MLP Result

From the above table, the original model has a high accuracy and AUC. Although Cluster Centroid under-sampling has higher AUC and higher Recall (which we think is the most important), its much lower accuracy is something we don't want. For the two over-sampling methods, the performance is good, but still worse than the original model. Therefore, if we will use the MLP as our model, I suggest just use the raw dataset while normalizing the features.

4 Conclusion and Potential Impact

To conclude, we put all the 3 models: Logistic Regression, XGBoost, and MLP together, along with the original data and 3 re-sampling methods: Cluster Centroid Under-sampling, Random Over-sampling, and SMOTE, and compare their accuracies, recalls, and AUC's.

	Metrics	Original	Cluster Centroid	Random Oversampling	SMOTE
LR	Accuracy	96.7347%	98.8390%	92.6150%	97.2590%
	Recall	87.7551%	85.0340%	93.8776%	89.7959%
	AUC	95.1968%	93.0927%	97.9608%	97.8451%
XGB	Accuracy	97.5961%	92.7121%	99.2697%	97.2590%
	Recall	89.7959%	87.7551%	88.4354%	89.7959%
	AUC	98.0967%	96.0731%	97.7857%	97.0353%
MLP	Accuracy	99.8432%	88.0973%	99.8034%	99.7355%
	Recall	86.3946%	94.5578%	85.0340%	83.6735%
	AUC	96.2743%	97.4141%	95.5979%	95.2461%

Figure 13: Result

If we just use the imbalanced data, MLP will have the highest accuracy, while XGBoost will have a higher Recall and AUC. Maybe after tuning the MLP more, we could arrive at a better performance. Therefore, I would recommend using XGBoost with original data, or if we have enough computational resources, we can tune the hyperparameters for MLP.

We know that the credit card transaction dataset is very large, where sometimes we might not be able to process all the data, so we need undersampling to reduce the number of non-fraud data. When we are using the Cluster Centroid Under-sampling, both XGBoost and MLP have low accurcies, maybe because these models are overfitting with the under-sampled small data, while the logistic regression has the highest accuracy, and acceptable Recall. Therefore, if we don't want so many data, than we can use Cluster Centroid Under-sampling and logistic regression, which will be faster and easier to explain.

If we could over-sample the data, when using XGBoost model, Random Over-sampling could be a good choice since it has a high accuracy and AUC, with acceptable Recall. When using logistic regression, SMOTE is a better choice.

So the following diagram is what we get so far a general process of how to deal with a raw highly imbalanced data:

Figure 14: Data Process

5 Code Avalibility and Future Step

The code of this project can be found on my Github. Due to the time constraints and computational resource limitations, the models are trained in Colab, and some models can be better tuned. Idealy, each model for each re-sampled data can be tuned to better performance, especially for XGBoost and MLP, and then we can see a more accurate comparison. Furthermore, each data is unique and has different performance on these models, so I will try some other datasets to see how our conclusion process works, or if there's some place to be adjusted for specific data.

6 Reference

1 Kaggle: https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud

2 Tiwari, Pooja, et al. "Credit card fraud detection using machine learning: a study." arXiv preprint arXiv:2108.10005 (2021).

3 Y. -P. Zhang, L. -N. Zhang and Y. -C. Wang, "Cluster-based majority under-sampling approaches for class imbalance learning," 2010 2nd IEEE International Conference on Information and Financial Engineering, Chongqing, China, 2010, pp. 400-404, doi: 10.1109/ICIFE.2010.5609385.

4 R. Mohammed, J. Rawashdeh and M. Abdullah, "Machine Learning with Oversampling and Undersampling Techniques: Overview Study and Experimental Results," 2020 11th International Conference on Information and Communication Systems (ICICS), Irbid, Jordan, 2020, pp. 243-248, doi: 10.1109/ICICS49469.2020.239556.

5 Fernández, Alberto, et al. "SMOTE for learning from imbalanced data: progress and challenges, marking the 15-year anniversary." Journal of artificial intelligence research 61 (2018): 863-905.