Lisbon School of Economics & Management
Universidade de Lisboa

MASTER

MASTER PROGRAM IN MANAGEMENT INFORMATION SYSTEMS

MASTER´S FINAL WORK

DISSERTATION

EVALUATING RESAMPLING TECHNIQUES FOR IMBALANCED CREDIT CARD FRAUD DETECTION USING XGBOOST

LI ZHU

JUNE - 2025

MASTER

MASTER PROGRAM IN MANAGEMENT INFORMATION SYSTEMS

MASTER´S FINAL WORK

DISSERTATION

EVALUATING RESAMPLING TECHNIQUES FOR IMBALANCED CREDIT CARD FRAUD DETECTION USING XGBOOST

LI ZHU

SUPERVISION:

PROF.CARLOS J. COSTA, PHD

JURY:
PROF. ANTONIO PALMA DOS REIS
PROF. RUI TRIGO
PROF. CARLOS J. COSTA

JUNE - 2025

ERRATUM

GLOSSARY

AdaBoost - Adaptive Boosting.

AUC - Area Under Curve.

BiGRU - Bidirectional Gated Recurrent Units.

BiLSTM - Bidirectional Long Short-term Memory.

CCF – Credit Card Fraud.

CIS - Computational Intelligence Society.

CNN - Convolutional Neural Network.

CRISP-DM - Cross-Industry Standard Process for Data Mining.

DRF - Distributed Random Forest.

FNN - Feedforward Neural Networks.

GAN - Generative Adversarial Network.

GBDT - Gradient Boosting Decision Tree.

GPU - Graphics Processing Unit.

IEEE - Institute of Electrical and Electronics Engineers.

ML - Machine Learning.

PR - Precision Recall.

RAM - Random Access Memory.

ROC - Receiver Operating Characteristic.

RUS - Random Undersampling.

SMOTE - Synthetic Minority Oversampling Technique.

XGBoost - eXtreme Gradient Boosting.

ABSTRACT

Credit card fraud detection is a key application of machine learning, but real-world fraud datasets are often highly imbalanced, with only a very small number of fraudulent transactions. The objective of this study is to explore how class imbalance affects model performance in fraud detection and to evaluate whether resampling strategies improve results. This paper investigates the impact of class imbalance on model performance and evaluates the effectiveness of several resampling techniques in improving prediction results. The XGBoost algorithm is used as a baseline classifier and trained on the original imbalanced data and data processed with various sampling strategies. Results show that XGBoost performs well even on the original imbalanced dataset after adjusting the class weights (scale_pos_weight). While SMOTE and SMOTETomek slightly improve precision, they reduce recall; NearMiss achieves the highest recall but has very low precision. This suggests that there may be a trade-off between identifying fraud and avoiding false positives.

This study emphasizes the importance of selecting a sampling strategy that considers not only technical performance but also business objectives. In many real-world applications, algorithm-level tuning methods may be simpler and more efficient than data-level resampling. The paper discusses the limitations of the algorithm and future work directions, including interpretability, data leakage risks, and the potential of threshold tuning or online learning strategies.

KEYWORDS: Financial Fraud Detection; Machine Learning; XGBoost; Resampling.

JEL CODES: C45; C52; G21; G32.

TABLE OF CONTENTS

TABLE OF FIGURES

Acknowledgments

First, I would like to thank Professor Carlos J. Costa for his patient guidance and constant encouragement. The thesis guidance across multiple time zones has given me a deeper understanding of a professional professor and scholar. Our discussions on the content of the thesis have also given me a lot of inspiration.

At the same time, I would like to thank my sister Lin ZHU, her concern helped me get through the weeks of insomnia.

1. Introduction

Information technology is developing rapidly day by day, which has revolutionized the financial industry, and is leading to an obvious shift from cash-based transactions to digital payment, including online banking, mobile payments, and credit card transactions. But meanwhile, there are also challenges born with the revolution. Credit Card Fraud (CCF) has emerged as one of the most important issues for banks, payment processors, and e-commerce platforms worldwide. It results in billions of dollars in annual loss (Tiwari et al., 2021). But for the safety of their customers' funds and their own good reputation, financial institutions will obviously not sit idly by. They also use the advanced achievements of information technology to arm themselves. In order to detect which credit card transactions are fraudulent transactions, they have developed automatic fraud detection systems. The common ones are Machine Learning (ML) models (Alarfaj et al., 2022). But building models, making it work well in reality, and being able to accurately identify fraudulent transactions is not as simple as just described.

One of the main challenges in credit card fraud detection is the class imbalance problem (Kalid et al., 2024, p. 23641). Class imbalance refers to the situation where the number of training examples for different classes in a classification task varies greatly. In most real-world transaction datasets, the number of legitimate transactions far exceeds the number of fraudulent transactions. This imbalance poses a serious problem for traditional machine learning classifiers, as classifiers tend to optimize overall accuracy by favoring the majority class. This often leads to learning biases that make the model poorly generalize to minority class patterns. And performance metrics can also be misleading when accuracy is the primary evaluation criterion. Because the proportion of fraudulent transactions is very small, a model may achieve a seemingly high accuracy simply by predicting all transactions as non-fraudulent transactions. However, this completely fails to detect real fraud cases, which seriously undermines the model's utility in real-world applications. Because in real-world applications, the model needs to identify a small number of fraudulent activities that cause financial losses and damage the company's reputation.

To address this, researchers have proposed various resampling techniques to balance the class distribution before model training (Mohammed et al., 2020). However, there remains limited consensus on which sampling strategies work best under different conditions.

This study aims to investigate how resampling techniques affect the performance of machine learning models for credit card fraud detection. Specifically, the objectives are:

1. To investigate the impact of class imbalance on the performance of financial fraud detection using XGBoost model.

2. To compare the performance of various resampling strategies, including undersampling (such as NearMiss), oversampling (such as SMOTE), and hybrid sampling (such as SMOTETomek).

3. To analyze the practical implications of resampling in real-world financial systems.

In this article, a machine learning-based fraud transaction detection system has been deployed on the IEEE-CIS Fraud Detection dataset, which is an imbalanced transaction dataset provided by Kaggle, to study how resampling techniques will affect the performance of machine learning models and the applicability of various sampling techniques in reality.

This study uses a systematic experimental approach to evaluate the impact of various resampling techniques on the fraud detection performance of the XGBoost model. First, the IEEE-CIS Fraud Detection dataset is divided into a training set and a test set. Then, the following steps are conducted:

1. Baseline model: The training set is initially used without any resampling, and the baseline XGBoost model is trained directly on the imbalanced data.

2. Resampling techniques: Four common resampling methods are applied to the training set to tackle the class imbalance problem of the dataset:

- The undersampling method NearMiss

- The undersampling method RUS

- The oversampling method SMOTE

- The hybrid method combining SMOTE and Tomek Links, SMOTETomek

3. Model training and evaluation: An XGBoost model is trained for each resampled version of the training data, resulting in a total of five models (including the baseline model).

4. Performance comparison: All models are evaluated on the same test set, and five key performance indicators are used: Accuracy, Precision, Recall, F1 score, and AUC. In addition, ROC and precision-recall (PR) curves are plotted for intuitive comparison.

This approach enables a comprehensive evaluation of how different resampling strategies affect model performance in the imbalanced fraud detection task. The full source code used in this study is available at Li Zhu (2025).

The rest of this paper is organized as follows: Section 2 will review the research background and explain the related works in detail. The proposed model, resampling techniques, and methodology will be elaborated on in Section 3. Section 4 is used to present the data results of the study. The discussion of the data results is placed in Section 5. Section 6 is the final conclusion of this paper.

2. Literature Review

2.1 Class Imbalance in Fraud Detection

CCF occurs when a credit card or account details are used by someone other than the cardholder to conduct an illegal transaction. Criminals often use this method to trick the cardholder into paying money into a bank account they control or to obtain goods and services (Bin Sulaiman et al., 2022). Typically, stolen, lost or counterfeit credit cards can lead to this fraud. With the increase in online shopping, card-not-present fraud is becoming more and more common (Tiwari et al., 2021).

Given the number of credit card transactions that occur every minute around the world, it is obviously impossible to rely on a manual review of the legality of each transaction. Therefore, the credit card fraud transaction detection system is used to automatically filter out those abnormal transactions. However, in actual transactions, the vast majority of transactions are legal, and fraudulent behavior is only a minor anomaly. Fraudsters will use various means to deliberately disguise themselves as normal users to avoid being identified by the detection system. The monitoring or detection systems of banks and trading platforms will promptly identify and block the fraudsters' means. As a result, the battle between fraud and anti-fraud has evolved into an endless marathon.

ML models are widely used on this battlefield (Tiwari et al., 2021). Financial institutions use them to learn, identify, and label fraudulent transactions. Faced with the ever-changing fraud methods, fraud transaction detection systems are also constantly evolving and fighting against them. Many scholars and experts have conducted relevant research on this topic.

Kalid et al. (2024) selected and studied 87 papers written in English and published from 2016 to 2023 that used public credit card fraud datasets to conduct academic research on credit card fraud transaction detection. They found that the problem of class distribution imbalance is one of the two main problems of the dataset, and all the research articles they screened have attempted to solve this problem. Because conventional machine learning models tend to be more biased towards the majority class, they cannot effectively detect the critical minority class that accounts for a small proportion (such as credit card fraud transactions that are small in proportion but highly harmful). In their study, they found that the three most commonly used technologies to solve this problem are deep learning and neural networks, ensemble learning, and sampling methods.

2.2 Resampling Techniques

Using resampling techniques to modify the data distribution in the training dataset is one of the common methods to solve the class imbalance problem (Muaz et al., 2020).

Resampling methods can be roughly divided into three categories: undersampling, oversampling, and hybrid sampling (Khushi et al., 2021). Undersampling aims to reduce the size of the majority class to match the minority class. This can improve balance but may lose important information. On the other hand, oversampling increases the number of minority class samples by copying existing instances or generating synthetic instances, but there is also a potential risk of overfitting. Hybrid sampling is a combination of undersampling and oversampling, which can avoid the defects of undersampling or oversampling to a certain extent and obtain a more ideal data structure.

NearMiss is a class of undersampling methods that aims to balance the class distribution by selectively removing majority class samples based on their distance from minority class samples. Specifically, NearMiss retains the majority class samples that are closest to the minority class, allowing the model to focus on more ambiguous or difficult areas of the decision boundary. There are multiple variants of NearMiss (e.g., NearMiss-1, NearMiss-2), each of which using a different strategy to determine which majority class instances to keep (Mqadi et al., 2021). This approach helps improve recall but may lose valuable information in the majority class.

The random undersampling (RUS) technique randomly removes samples from the majority class to balance the entire dataset, which is simple and fast (Prusa et al., 2015). However, this method is completely random, so it does not take into account the data distribution of the dataset itself, so characteristic samples may be discarded when removing samples, resulting in potential information loss (Hasanin & Khoshgoftaar, 2018). To solve this problem, I applied Tomek Links after RUS to further clean up the boundaries between classes. Tomek Links exist between two samples that are adjacent to each other and belong to different classes. Removing such links helps reduce overlapping and ambiguous samples, thereby improving the classifier's performance (Mittal et al., 2025).

Synthetic Minority Oversampling Technique (SMOTE) is an oversampling technique that synthetically generates new instances of the minority class by interpolating between existing minority class samples and their nearest neighbors. Instead of simply duplicating existing observations, SMOTE identifies k nearest neighbors for each minority class instance in the feature space. For each new synthetic sample, the algorithm randomly selects one or more of these nearest neighbors and generates a new data point along the line segment connecting the original instance and the selected nearest neighbor, which is scaled with a random value between 0 and 1. The core assumption behind SMOTE is that similar instances (i.e., instances that are close to each other in feature space) belong to the same class and have similar properties (Chawla et al., 2002). Therefore, SMOTE provides a more diverse and generalized set of training samples by interpolating in feature space instead of duplicating data in data space. This generally improves the performance of classifiers on imbalanced datasets (Almazroi & Ayub, 2023). However, a key limitation of SMOTE is that synthetic instances may not always reflect the true underlying data distribution. Since it assumes a linear relationship between neighboring samples, it may introduce ambiguous or noisy data points, especially in areas where classes overlap or feature distributions are highly nonlinear. This can sometimes increase the risk of overfitting or reducing precision, thereby degrading the performance of the model (Tarawneh et al., 2022).

SMOTETomek is a hybrid sampling technique that combines the advantages of SMOTE (oversampling) and Tomek Links (undersampling) to improve the diversity of minority class samples and the quality of class boundaries (Wang et al., 2019). After applying SMOTE to generate synthetic minority class instances, the Tomek Links algorithm is used to further optimize the dataset to identify and remove boundary instances that may introduce noise or class overlap. As mentioned earlier, a Tomek Link exists between a pair of samples from different classes that are adjacent to each other. Such sample pairs are considered ambiguous and are usually located near the decision boundary. In SMOTETomek, the majority class instances in each Tomek Link are removed to clean up the boundaries and reduce class ambiguity (Shabrina Assyifa & Luthfiarta, 2024). On the one hand, SMOTETomek improves recall by adding synthetic minority class samples, and on the other hand, it also improves precision and generalization by removing majority class samples that may be noisy or overlapping. As a result, SMOTETomek generally achieves a better balance between sensitivity (recall) and specificity (precision), making it particularly effective for real-world fraud detection, medical diagnostics, or any field where both types of errors (false positives and false negatives) carry significant costs.

2.3 Machine Learning Algorithms

The problem of data class imbalance has long attracted the attention of researchers. A wide range of ML algorithms has been applied.

Many scholars have tried to use different classifiers with resampling methods to explore ways to solve the problem of data imbalance. For example, Ghosh et al. (2024) revealed the similarities and differences between deep neural networks and traditional machine learning models under class imbalance through systematic literature review and empirical experiments, and pointed out the limitations of existing solutions. Through literature review, they confirmed that although deep learning has stronger feature learning capabilities, class imbalance still causes the model to be biased towards the majority class. However, current measures seem to have defects, such as: traditional interpolation methods are difficult to apply to image data, and resampling methods are usually only applied at the small batch level, which will affect the efficiency of model training.

Muaz et al. (2020) pointed out that the cost of misclassification is significantly asymmetric: misclassifying normal transactions as fraud (false positive) will cause management costs to the company, while missing fraudulent transactions (false negative) will lead to direct financial losses. Therefore, this study emphasizes that recall should be used as a core indicator to maximize the recognition rate of fraudulent transactions (true positive rate). To address the problem of class imbalance, the author systematically compared the combined effects of four sampling strategies and four classifiers. Experiments show that SMOTE oversampling performs best; the Distributed Random Forest (DRF) classifier achieves the highest recall rate (0.81) and precision rate (0.86) on the SMOTE dataset, proving that synthesizing minority class samples can effectively improve the model's capture of fraud patterns.

Meng et al. (2020) studied the performance of the XGBoost algorithm on the original dataset, undersampled dataset, and SMOTE dataset based on the real online transaction data of an Internet financial institution. The credit fraud samples in the original dataset only accounted for 0.172% of all samples. After oversampling using the SMOTE algorithm, the XGBoost classifier achieved convincing results of 0.9 recall and 0.98 AUC. They believe that SMOTE balances the original dataset and improves the stability and generalization ability of the classifier.

In order to solve the problem of class imbalance, Cheah et al. (2023) innovatively introduced two hybrid technology solutions, SMOTE+GAN and GANified-SMOTE. They used FNN, CNN and FNN+CNN as classifiers. The research results highlight the powerful performance of GANified-SMOTE, a hybrid sampling technology, especially when combined with the proposed FNN+CNN classifier, which is particularly outstanding in improving the F1 score of fraud data. The high F1 score indicates that this method is able to identify a large number of fraudulent transactions and reduce the misclassification of legitimate transactions. In addition, this study also emphasizes the importance of the classifier hyperparameter settings. Reasonable settings will have a positive impact on classification performance. Singh et al. (2022) used several resampling methods and different classifiers and compared multiple performance indicators obtained after model training, such as accuracy, recall, K-fold cross validation, AUC-ROC curve, and execution time. They found that for ensemble classification models such as AdaBoost, XGBoost, and Random Forest, the method of oversampling followed by undersampling performed well. Rubaidi et al. (2022) applied and experimented with two undersampling techniques, Random Undersampling and NearMiss, and three oversampling techniques, Random Oversampling, SMOTE, and BorderLine SMOTE, and used different evaluation metrics to evaluate the classifiers. They concluded that the Random Forest model with Random Oversampling achieved an accuracy of 99.3%. This is because the Random Forest classifier categorizes samples based on the majority vote of multiple decision trees. They also observed that oversampling achieved better results than undersampling for different classifiers. Mittal et al. (2025) also applied SMOTE and Tomek Link for resampling in their study on the impact of imbalanced datasets on classification machine learning models. Experimental data showed that the performance of some classifiers was indeed improved.

But not everyone thinks that resampling techniques are the only effective method, and they use different methods to improve the performance of models. The research of Zhang et al. (2020) shows that feature engineering and visualization also have a positive effect on improving model performance. Dedy Trisanto et al. (2021) proposed an improved Focal Loss method that does not require traditional data preprocessing steps such as sampling or outlier detection. This method is based on weighted binary cross-entropy and introduces an imbalance parameter to adjust the loss function to increase the model's attention to minority classes in the data set, achieving excellent performance of 0.88 precision and 0.87 recall.

Ileberi et al. (2021) implemented several machine learning algorithms for credit card fraud detection and paired them with AdaBoost technology to improve the performance of classifiers. Experimental results show that the use of the AdaBoost algorithm has a positive impact on some machine learning methods, with a test accuracy of more than 99%. Tarawneh et al. (2022) criticized the direction of using oversampling methods to solve class imbalance problems in sensitive fields. They believe that current oversampling methods are misleading. Their verification on real-world datasets shows that current common oversampling methods generate wrong samples, which are likely to cause prediction errors, making them unreliable in practical applications. Synthesized samples may not accurately represent minority groups, leading to potential failures in practice. Therefore, they recommend avoiding the use of oversampling techniques in sensitive applications such as security and healthcare.

Through research, Carvalho et al. (2025) believe that no resampling method can guarantee excellent performance in all application scenarios. Balancing the classes in a dataset does not inherently mitigate all biases in the data. When choosing a resampling method, one should always consider not only the specific application and user preferences but also the data characteristics of different categories in the dataset. Brandt & Lanzén (2020) also found that as the degree of class imbalance changes, no resampling technique always plays a positive role in contributing to performance improvement. Resampling techniques can sometimes improve the overall performance of the model, but sometimes give results similar to the unprocessed data model, or even reduce performance. de la Bourdonnaye & Daniel (2022) believes that resampling methods that can complete processing in a reasonable time are more worthy of consideration, so they also take computing time into consideration. Based on this criterion, they use selected resampling methods on large-scale data sets to compare the impact of resampling on the performance of different models. The results show that resampling technology does not show higher efficiency and performance.

3. METHODOLOGY

This study adopts the CRISP-DM (Cross-Industry Standard Process for Data Mining) framework to guide the experimental workflow. The CRISP-DM process provides a systematic approach to building data mining projects, including stages such as problem understanding, data preparation, modeling, evaluation, and result interpretation (Costa, & Aparicio, 2020, Costa & Aparicio, 2021).

With this framework, this study aims to investigate how class imbalance impacts the performance of machine learning models in credit card fraud detection, utilizing the XGBoost algorithm. The IEEE-CIS fraud detection dataset is characterized by an imbalance between fraudulent and legitimate transactions, and the dataset is employed to replicate real-world scenarios.

Six experiments are conducted: one with the original dataset and four with different resampling techniques — NearMiss, RUS combined with Tomek Links, SMOTE, and SMOTETomek. The sixth is an additional experiment using SMOTE on a small sample set. Each model is evaluated on a consistent test set using multiple performance metrics, including accuracy, precision, recall, F1, AUC, and diagnostic visualization tools such as ROC curves and PR curves.

This methodology is closely aligned with the research objectives: assessing the impact of class imbalance, evaluating common resampling techniques, and explaining their practical implications for real-world financial systems.

3.1 Dataset and Tools

The dataset used in the experiments in this article is the IEEE-CIS Fraud Detection dataset. The IEEE-CIS Fraud Detection dataset was provided by Vesta in 2019 for Kaggle's "IEEE-CIS Fraud Detection" competition (IEEE Computational Intelligence Society, 2019), which aims to use machine learning models to identify fraudulent online transactions in consumer transaction data. The dataset comes from e-commerce transaction records collected by Vesta in the real world, containing more than 590,000 transaction records, of which about 3.5% are marked as fraudulent transactions, highlighting the serious class imbalance inherent in real-world fraud detection scenarios.

The dataset is mainly divided into two parts:

-Transaction data: contains details such as transaction amount, product code, payment card information, and anonymous features marked as V1 to V339 for privacy protection purposes.

-Identity data: contains information related to device type, browser details, and other user-specific attributes.

The two datasets are linked by a common identifier "TransactionID" and can be combined for comprehensive analysis.

The main goal of using this dataset is to develop machine learning models that can predict the possibility of transaction fraud. Given the complexity of this dataset (including high dimensionality and class imbalance), it is an excellent benchmark for testing various sampling techniques and classification algorithms, and is widely used by researchers and machine learning enthusiasts.

This paper will use this classic, class-imbalanced, real-world fraud transaction dataset to study how various sampling techniques affect the accuracy and usability of machine learning models for abnormal transaction detection.

Since this dataset is a competition dataset, there are no fraud labels in the official test set, so it is impossible to test the performance of the model training. Therefore, the training set on the official website is used as the original dataset to ensure that the performance of the model has intuitive data charts for comparison. In the following text, this dataset will be divided into training set and test set in a ratio of 8 to 2.

All experiments in this study were conducted using Kaggle Notebooks, a cloud-based Jupyter Notebook environment widely used in data science and machine learning research. The platform provides pre-installed libraries and efficient GPU acceleration, enabling rapid training and evaluation of models on large-scale datasets. At the same time, since the IEEE-CIS Fraud Detection dataset is a competition dataset of Kaggle, it does not need to be downloaded and can be directly imported.

The hardware and system specifications are as follows:

- GPU accelerator: NVIDIA Tesla P100

- RAM: 30 GB RAM

- Storage space: 57.6 GB available disk space

- Python version: Python 3.10

In order to solve the class imbalance problem, the imbalanced learning library (version 0.10.1) was installed in the Notebook environment using the "!pip install irrebalanced-learn==0.10.1" command.

The library provides a variety of resampling techniques, such as NearMiss, RUS, Tomek Links, SMOTE, and SMOTETomek, which are essential for the comparative analysis in this study. The resampling techniques used are described in detail later.

The full set of tools and libraries used in the experimental process include:

- XGBoost: for building and training gradient-boosting classification models

- imbalanced-learn: for applying various resampling techniques to imbalanced datasets

- Scikit-learn: for model evaluation and metric calculation (e.g., precision, recall, F1, AUC)

- Pandas and NumPy: for data manipulation and preprocessing

- Matplotlib and Seaborn: for visualizing evaluation metrics, such as ROC curves, PR curves, and bar charts

3.2 Model Description

eXtreme Gradient Boosting (XGBoost) is an ensemble learning algorithm based on the idea of Gradient Boosting Decision Tree (GBDT), proposed by Chen & Guestrin in 2016. It has become one of the most widely used algorithms in machine learning competitions and practical applications (such as fraud detection, recommendation systems, and search ranking).

The GBDT algorithm is an ensemble learning method that combines multiple weak learners (usually shallow decision trees) in a sequential manner. Each new decision tree is trained to minimize the error caused by the previous tree ensemble and uses gradient descent on the loss function.

Compared with traditional GBDT, XGBoost contains the following key improvements:

- Regularization: XGBoost adds L1 and L2 regularization terms in the objective function, which helps prevent overfitting and improves model generalization ability.

- Tree pruning and parallelization: Although the tree is still built sequentially, XGBoost performs optimized parallel calculations during the node-splitting process of each tree, which significantly improves efficiency.

- Column subsampling: Similar to random forest, XGBoost randomly extracts a subset of features when building trees, which reduces overfitting and speeds up computation.

- Missing value handling: XGBoost automatically learns the best direction for missing values, making it robust to incomplete data.

- Shrinkage and learning rate: A shrinkage factor is applied after adding each tree to slow down the learning process and improve performance (Chen & Guestrin, 2016).

Considering the large number of missing values in the selected dataset, and XGBoost has advantages in both prediction accuracy and computational efficiency, I chose it as the model for the experiments in this paper.

The following hyperparameters were set in the XGBoost model in this article:

- n_estimators=500: This is the number of decision trees in the ensemble. Generally, more trees give better results, but are slower to compute.

- max_depth=9: This is the maximum depth of each tree. Deeper trees can capture more complex patterns, but increase the risk of overfitting.

- subsample=0.9: This is the proportion of training instances used in each boosting round. Using only 90% of the instances can prevent overfitting.

- colsample_bytree=0.9: This is the proportion of features used when building each tree. Each tree randomly uses 90% of the feature columns to avoid over-reliance on certain features, which also helps reduce overfitting.

- missing=-999: All missing values are filled with -999 to avoid errors.

- device='cuda': This means that the model is trained on the GPU provided by the kaggle notebook, which can speed up training.

- reg_alpha=0.1 and reg_lambda=1: These are L1 and L2 regularization terms, which are used to control model complexity and prevent overfitting.

- eval_metric='auc': The model is evaluated based on the AUC of the ROC curve, which is suitable for imbalanced classification problems.

- scale_pos_weight=19: (Only for the original imbalanced dataset) Balance the positive and negative classes by increasing the weight of the minority class.

3.3 Evaluation Metrics

This part will introduce the metrics used to measure the effectiveness of the fraud detection model.

The first is accuracy, which refers to the proportion of samples predicted correctly by the model to the total number of samples, and the mathematical formula is:

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

TP is a true positive example (predicted as fraud, and it is fraud), TN is a true negative example (predicted as non-fraud, and it is non-fraud), FP is a false positive example (predicted as fraud, and it is non-fraud), and FN is a false negative example (predicted as non-fraud, and it is fraud). This metric is very intuitive, but in severely unbalanced data, accuracy is easy to mislead. For example, 99% of transactions are normal. Even if the model predicts all transactions as "non-fraud", accuracy can be as high as 99%, but the model is actually completely ineffective.

The second is precision, which is the proportion of all transactions predicted as "fraud" that are actually fraud. The formula is:

$$Precision = \frac{TP}{TP + FP}$$

It is usually used to measure the probability of false positives. The higher the precision, the fewer normal users are hurt by mistake and the less impact on user experience. It is especially suitable for scenarios where the cost of false positives is high, such as when a user's transaction is locked or the account is frozen because it is misjudged as a fraudulent transaction.

The third is recall, which refers to the proportion of all real fraudulent transactions that are correctly identified by the model. The formula is:

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

It is generally used to measure the false negative rate. The higher the recall, the stronger the model's ability to identify fraudulent transactions. This is very critical in fraud detection, because missing real fraud may cause financial losses.

The fourth is F1-score, which is the harmonic mean of precision and recall and is a comprehensive balance indicator of the two. The formula is:

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall}$$

When it is necessary to strike a balance between avoiding false positives and catching as many frauds as possible, the F1-score is a key indicator. In model evaluation, the F1-score is a common comprehensive indicator for imbalanced data sets.

The fifth is the Receiver Operating Characteristic (ROC) Curve, which is a curve drawn with the false positive rate (FPR) as the horizontal axis and the true positive rate (TPR, i.e. recall) as the vertical axis. The calculation formula is: TPR = TP / (TP + FN), FPR = FP / (FP + TN). In fraud detection, it is used to observe the performance of the model at different thresholds. Generally, the closer the curve is to the upper left corner, the better the model performs.

The sixth is the Area Under Curve (AUC), which is the area under the ROC curve and measures the model's ability to distinguish between positive and negative classes. Its value range is between 0.5 and 1, where 0.5 means that the model is no different from random guessing and 1 represents perfect prediction. In other words, the closer the AUC is to 1, the more capable the model is at distinguishing fraud from non-fraud. When the samples are imbalanced, AUC is a more reliable overall performance indicator than accuracy.

The last is the Precision-Recall(PR) Curve, which is a curve drawn with the model's recall as the horizontal axis and precision as the vertical axis. The PR curve can directly reflect the model's true ability to detect fraud. By comparing the PR curves of multiple models, researchers can determine which model can recall more fraud while maintaining high precision.

3.4 Experimental Procedure

The experiment follows a structured data preprocessing, model training, and performance evaluation pipeline. The specific steps are as follows:

Step 1: Dataset Import and Preliminary Analysis

After importing the data, a preliminary analysis was performed to check the distribution of the target variable.

Step 2: Feature Engineering and Data Preparation

After loading the data, multiple feature engineering techniques were applied. Non-numeric columns were encoded using label encoding to make the dataset suitable for XGBoost. The dataset was then split into training and test sets.

Step 3: Baseline Model Training without Resampling

As a baseline, we trained the training set without any resampling techniques. An XGBoost classifier was trained on this imbalanced dataset and the scale_pos_weight parameter was used to handle the class imbalance. The model was then evaluated on the test set.

Step 4: Model Training with Four Resampling Techniques

To address the class imbalance issue, we apply three resampling techniques to the training set:

- NearMiss
- RUS with Tomek Links
- SMOTE

For each resampled dataset, we train and evaluate a separate XGBoost model (with the same hyperparameters but without scale_pos_weight) and evaluate it on the same test set. This allows for a direct comparison of model performance under different resampling strategies.

Step 5: Fair Comparison of SMOTETomek and SMOTE Results

Due to the large size of the original dataset (~590,000 rows), applying SMOTETomek to the full dataset causes the Kaggle Notebook environment to crash. To mitigate this issue, a subset of 120,000 samples was randomly extracted from the dataset for SMOTETomek-based training and testing. The same 120,000 sample subset was then used to train and test the SMOTE-based model. The XGBoost classifier settings used in this step are the same as those in step 4, to compare the impact of SMOTETomek and SMOTE on the performance of the classifier on a small sample subset.

Step 6: Performance evaluation and metric comparison

Use common classification metrics (accuracy, precision, recall, F1 score, and AUC) to evaluate model performance in all five experimental settings. The results are visualized using bar charts and ROC/PR curves for direct comparison.

3.4.1 Preliminary Analysis

After loading and merging the dataset, we first observe the ratio of positive and negative samples and the missing values. From Figure 1, we can see that fraudulent transactions account for about 3.5% of the dataset. The dataset has 590,540 rows and 433 columns, of which 414 columns have missing values.

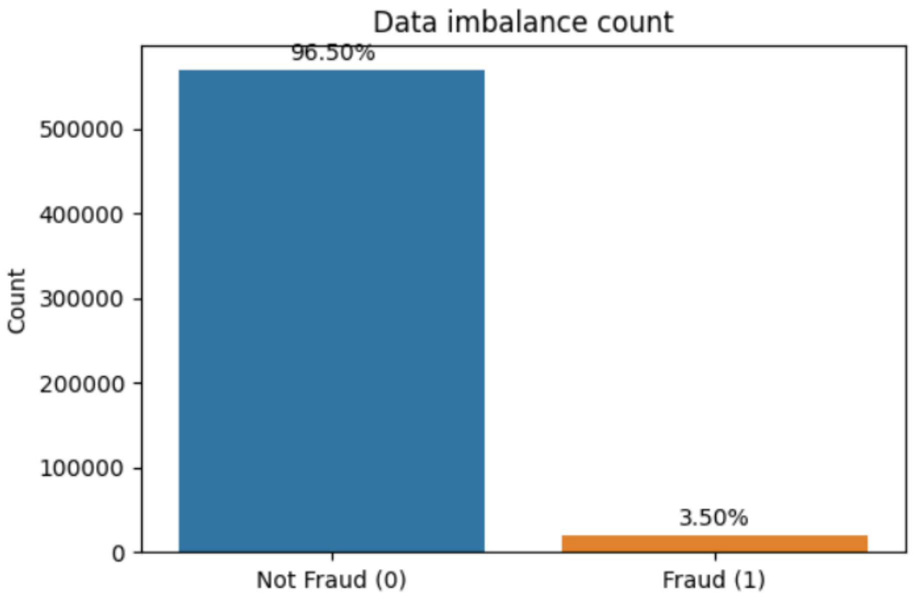

FIGURE 1 – Data Imbalance Count.

Next, let's look at the distribution histogram of TransactionAmt, with the horizontal axis being the transaction amount and the vertical axis being the frequency. As can be seen from Figure 2, the distribution of TransactionAmt is highly right-skewed, indicating that most transaction amounts are relatively small, while a small number of high-value transactions extend to the tail. This imbalance may cause the model to underestimate high-value fraudulent transactions, so we use a logarithmic transformation to reduce the impact of extreme values and normalize the distribution. The results after logarithmic transformation are shown in Figure 3.

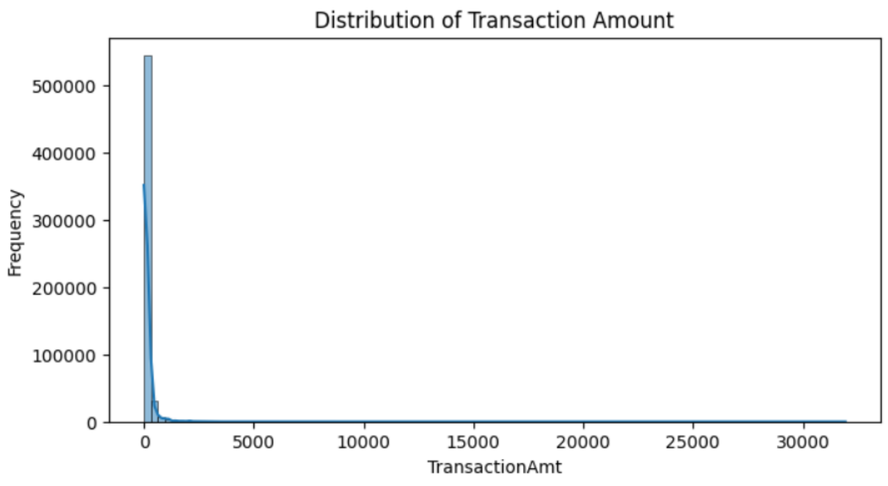

FIGURE 2 – Distribution of Transaction Amount.

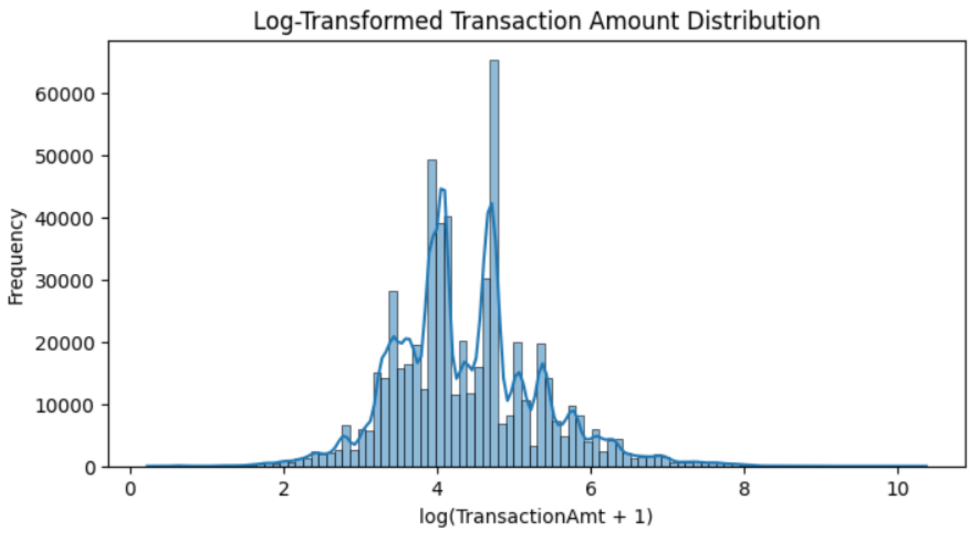

FIGURE 3 - Distribution of Log-Transformed Transaction Amount.

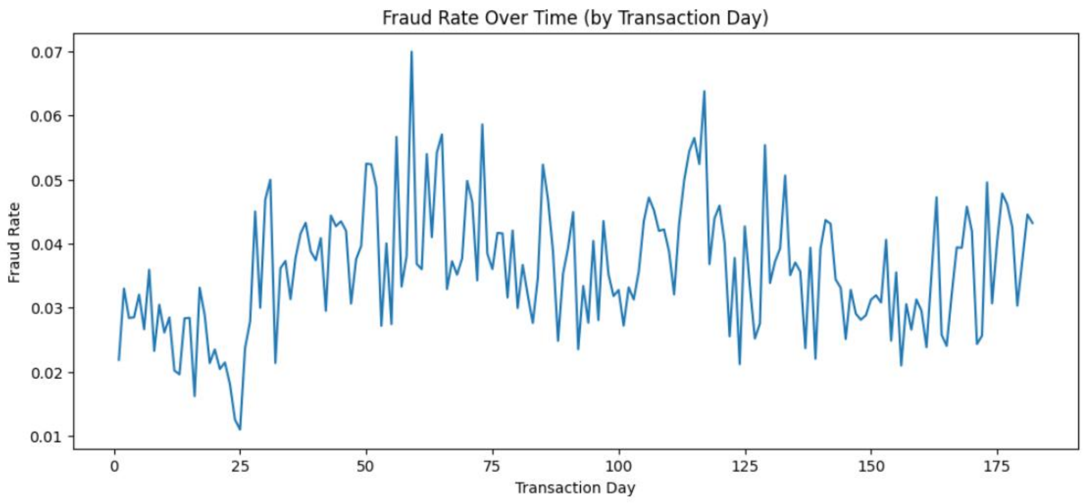

FIGURE 4 - Fraud Rate Over Time

Figure 4 is a graph of fraud rates over time. As can be seen in the figure, the fraud rates fluctuate over time, with increased fraud activity on certain dates. This suggests that the incidence of fraud is not evenly distributed over time, which may indicate the presence of coordinated attack cycles or time patterns.

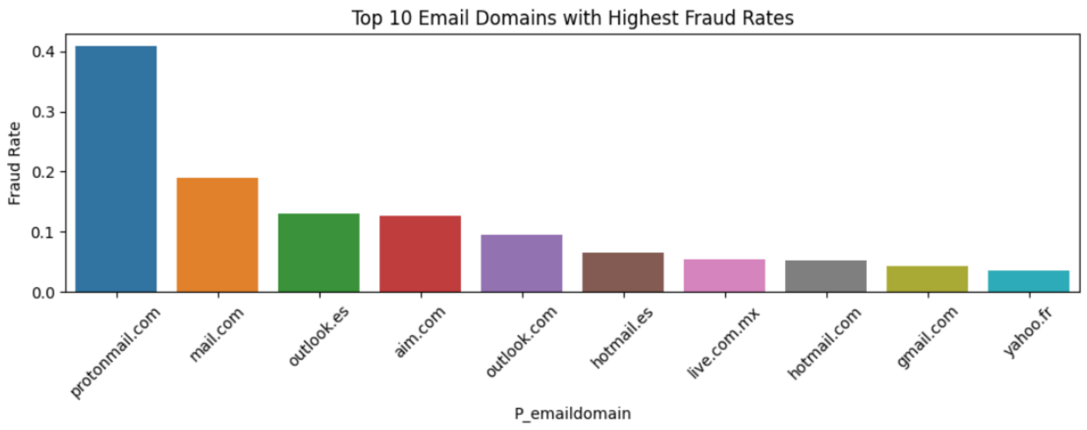

FIGURE 5 - Top 10 email domains with the highest fraud rates

Figure 5 is a bar graph of the relationship between email domain names and fraud rates, listing the top 10 email domains with the highest fraud rates.

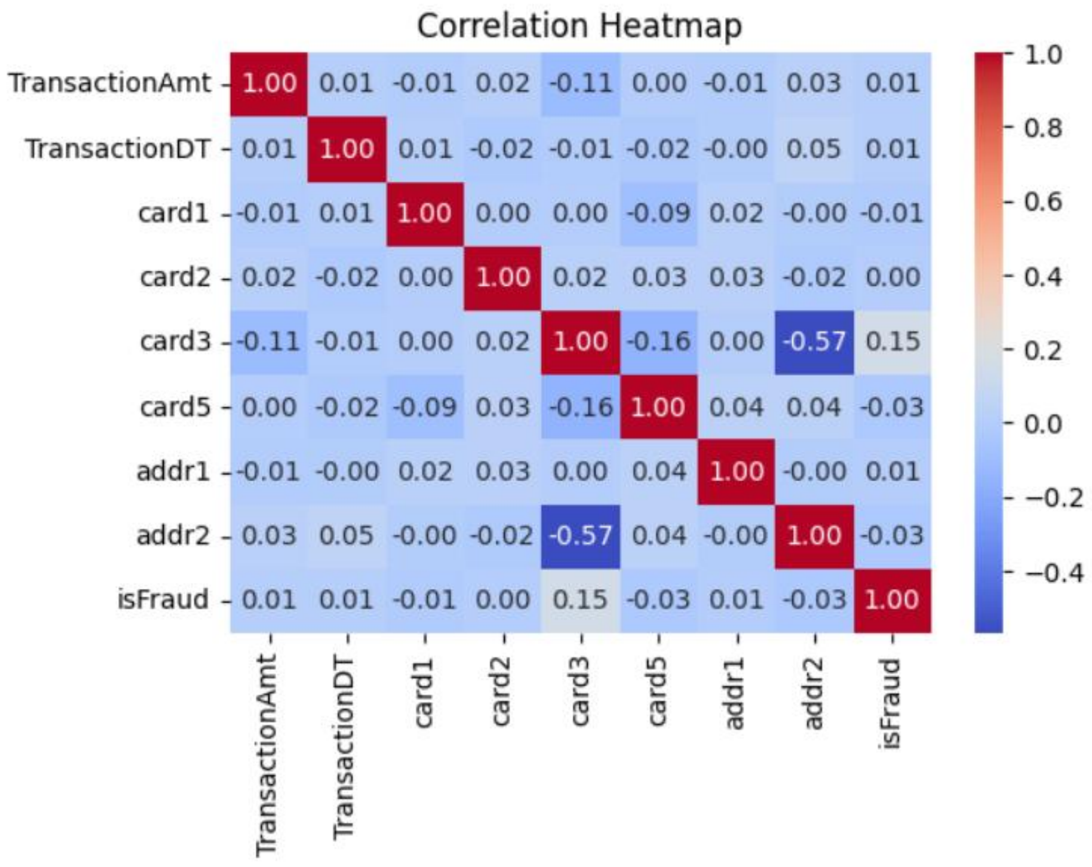

FIGURE 6 - Correlation Heatmap

Figure 6 shows a heatmap of the correlation between some numerical features and fraud. The correlation between these features and the target label "isFraud" is minimal, indicating that a few features alone are not strong predictors of fraudulent activity. This highlights the need for more sophisticated feature engineering and model-based approaches.

3.4.2 Feature Engineering

To improve model performance and provide more signals for fraud detection, several feature engineering techniques were applied before model training.

First, the dataset was divided into two subsets, X and Y. X contains all transaction features, and Y only contains information on whether it is fraudulent. All feature engineering techniques were performed on the subset X.

Second, TransactionAmt was log-transformed to reduce right skewness and stabilize variance. In addition, a missing value count feature (null value) was created for each row to capture potential information in the sparse pattern of the data.

Next, frequency encoding was applied to high-cardinality categorical variables such as card1, card2, addr1, addr2, P_emaildomain, ProductCD, card4, card6, and DeviceType. These frequency-based features provide contextual information about how common each value is in the dataset.

Aggregate features (including mean and standard deviation) were also created using TransactionAmt grouped by card1 and addr1 to capture user-level transaction patterns.

Several time-based features are derived from TransactionDT, such as transaction hour, date, weekday, hour interval, and whether the transaction occurred on the weekend. These features help identify behavioral or temporal patterns associated with fraud.

The email domain field is simplified to general categories (e.g., Gmail, Yahoo, Hotmail) and then label encoded. In addition, the DeviceInfo field is parsed to extract device brand information, and a binary variable is_mobile is created based on DeviceType.

Finally, combined categorical features (e.g., card1_card4 and addr1_email) are constructed and frequency encoded to capture potential interactions between user identifiers and email addresses.

These feature engineering steps aim to enrich the dataset by deriving variables to enhance the model's ability to detect fraudulent transactions while maintaining reasonable computational efficiency.

Since XGBoost requires all features to be numeric, all remaining non-numeric categorical columns after feature engineering are encoded using LabelEncoder. This ensures compatibility with the model while not affecting previously encoded numeric or aggregate features. The dataset is then split into training and test sets in a ratio of 8 to 2. To speed up computation, cupy is used to convert the test set to a GPU-accelerated format. This completes the data preparation.

It should be emphasized here that the core of this study is to compare the impact of different sampling methods on XGBoost performance rather than pursuing absolute performance indicators. Feature engineering is done before the training and test set splits, although it may introduce slight information leakage, but mainly to ensure a consistent benchmark for all sampling methods.

In future work, this could be improved by engineering features only on the training data and then applying them to the test set to strictly avoid leakage.

3.4.3 Model Training and Evaluation

As a baseline, the XGBoost model was first trained on the original imbalanced dataset without applying any resampling techniques. The "scale_pos_weight" parameter was adjusted to address class imbalance, and the test set was used to evaluate performance. The results serve as a reference point for evaluating the effectiveness of various resampling methods.

To evaluate the impact of resampling, then the training set were modified using the following techniques: NearMiss, RUS + Tomek Links, SMOTE, and SMOTETomek.

After resampling, we used each modified training set to train a new XGBoost model with the same hyperparameters (without "scale_pos_weight" due to adjusted class balance). We then evaluated the trained models on the test set.

Due to the high computational cost and memory footprint of SMOTETomek on the full dataset (over 590,000 records), we applied this hybrid approach to a randomly selected subset of 120,000 samples. To ensure comparability, SMOTE was also applied to the same subset of 120,000 samples. This controlled experiment allowed us to directly compare SMOTE and SMOTETomek under equivalent conditions.

For each model, we calculated the following evaluation metrics on the test set: accuracy, precision, recall, F1 score, and AUC.

All results are stored and visualized using the following formats:

- Bar charts are used to compare each metrics under different sampling strategies.

- ROC curves are used to compare the true positive rate and false positive rate.

- PR curves are used to evaluate the performance of detecting rare fraud cases.

4. RESULTS

Based on the XGBoost model, the experimental data obtained without sampling and using resampling technology are summarized in Table 1.

TABLE I

PERFORMANCE COMPARISON

	Accuracy	Precision	Recall	F1	AUC
No Sampling	0.97828258	0.65396700	0.80571013	0.7219512	0.97235173
NearMiss	0.59647949	0.07562107	0.93830147	0.1399621	0.84081919
RUS+tomek	0.92783723	0.30752367	0.84853617	0.4514385	0.95526443
SMOTE	0.98281234	0.93218249	0.54875393	0.6908315	0.95838725
SMOTETomek (120000 samples)	0.98017069	0.90174966	0.48632954	0.6318767	0.94173635

The model trained on the original imbalanced dataset achieved high AUC (0.972) and accuracy (0.978), indicating good overall discrimination capability. However, the precision (0.653) is relatively low, reflecting a high false positive rate. The recall (0.805) suggests that a significant portion of the minority class was still correctly identified despite the imbalance.

NearMiss produced the highest recall (0.938), meaning it was able to detect nearly all fraudulent transactions. However, this came at the cost of an extremely low precision (0.075) and F1 score (0.139), indicating that the majority of the positive predictions were incorrect. Additionally, the overall accuracy dropped dramatically to 0.596. This suggests that the aggressive undersampling severely disrupted the original data distribution, leading to poor model generalization.

The Random Undersampling with Tomek Links method showed more balanced results. Precision (0.307) and recall (0.848) were both moderate, with an AUC of 0.955. This method retained more data structure than NearMiss, reducing noise while removing ambiguous instances. As a result, it offered a good trade-off between performance metrics.

SMOTE achieved the highest precision (0.98), meaning the model was very conservative in labeling fraudulent transactions and rarely misclassified normal ones. However, the recall dropped to 0.548, indicating that many actual fraud cases were missed. This trade-off suggests that while SMOTE is effective at generating synthetic minority samples, it may cause the model to under-identify rare events in order to avoid false positives.

The comparison of the measurement results of each model is shown in Figures 7 to 13.

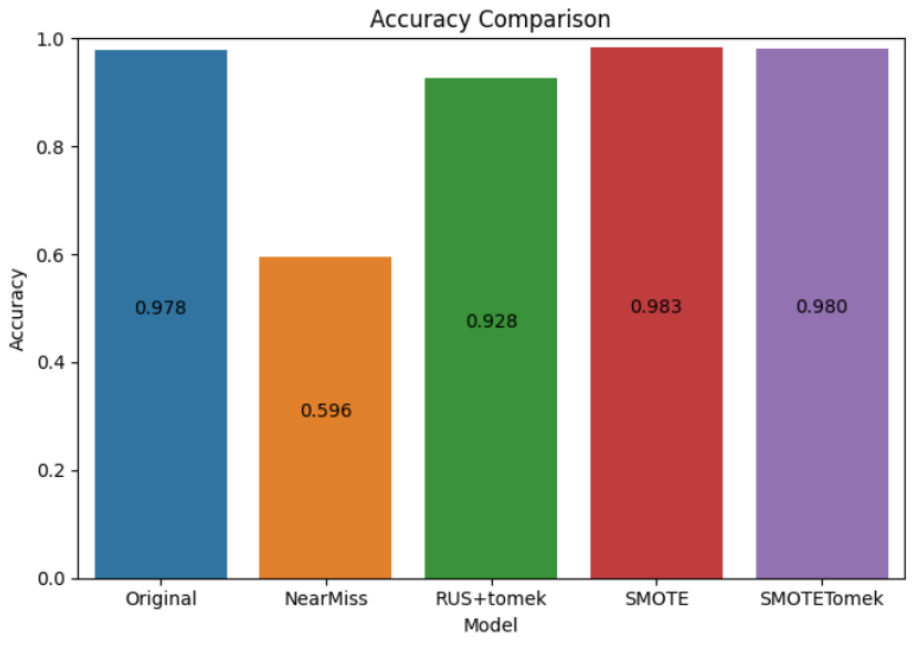

FIGURE 7 - Accuracy Comparison

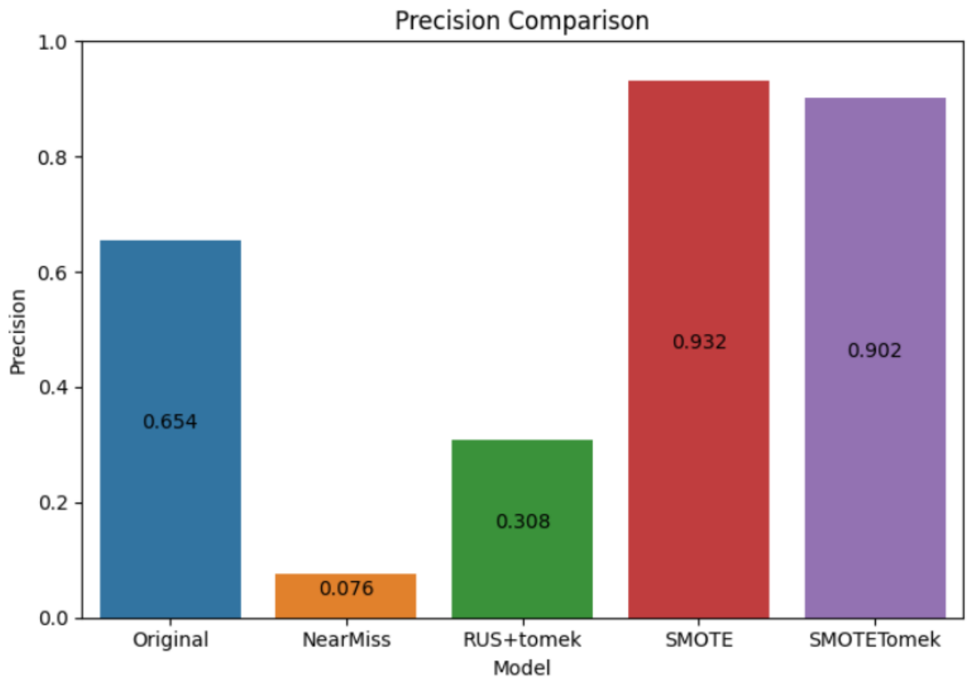

FIGURE 8 - Precision Comparison

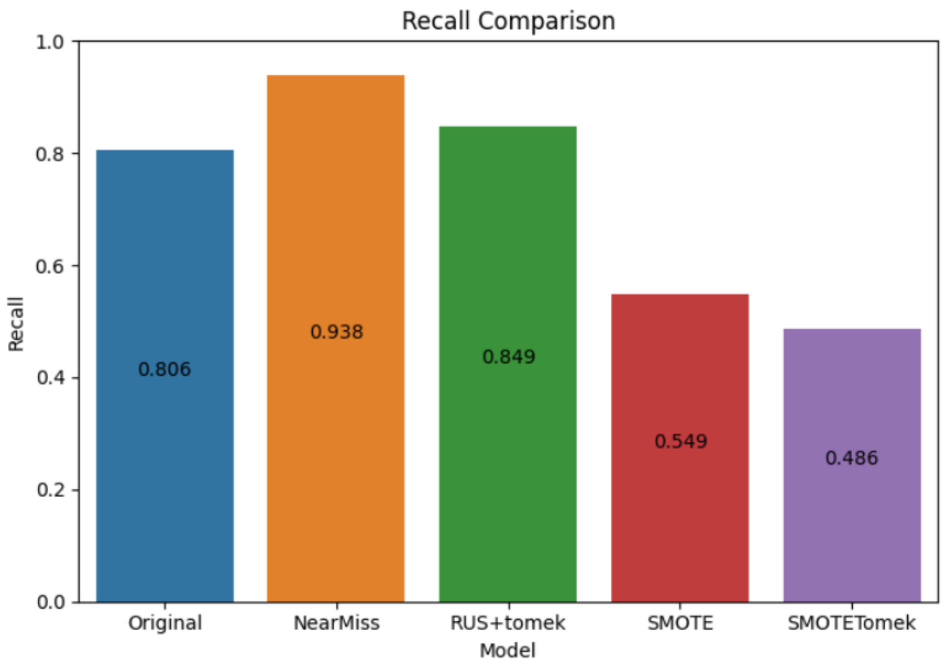

FIGURE 9 - Recall Comparison

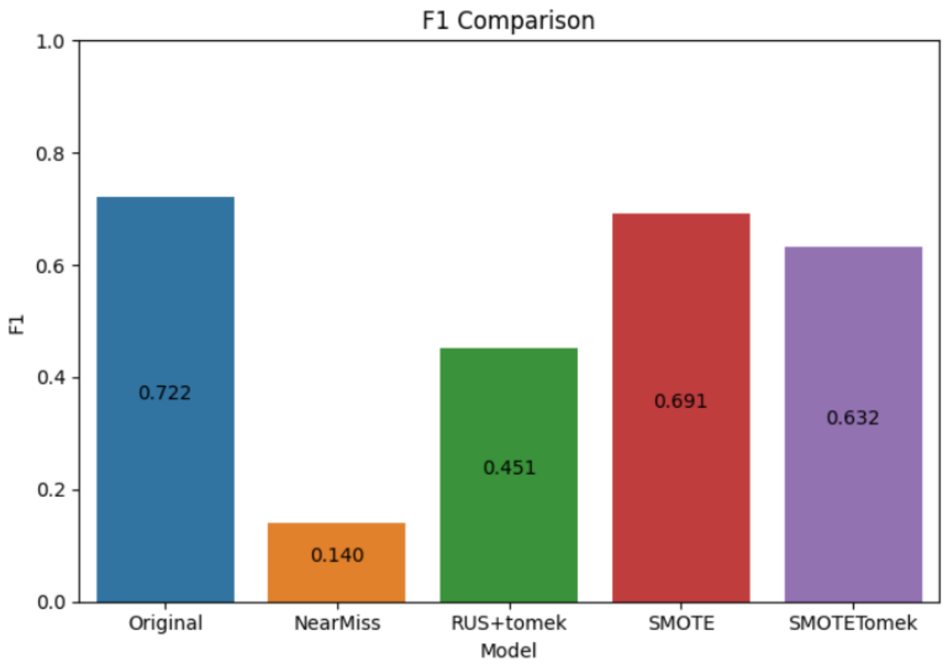

FIGURE 10 - F1 Comparison

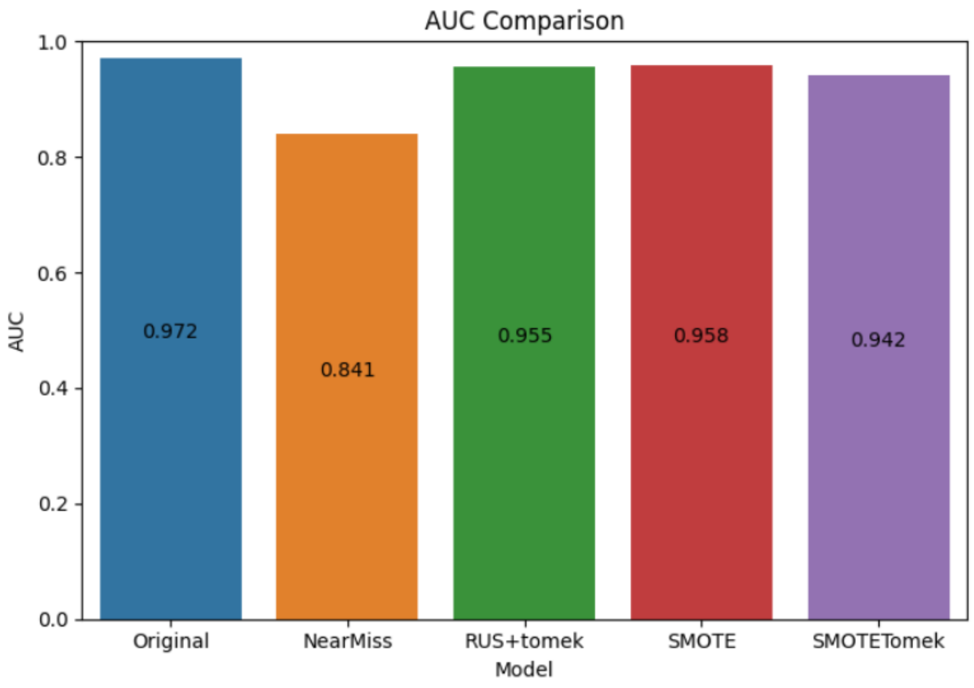

FIGURE 11 - AUC Comparison

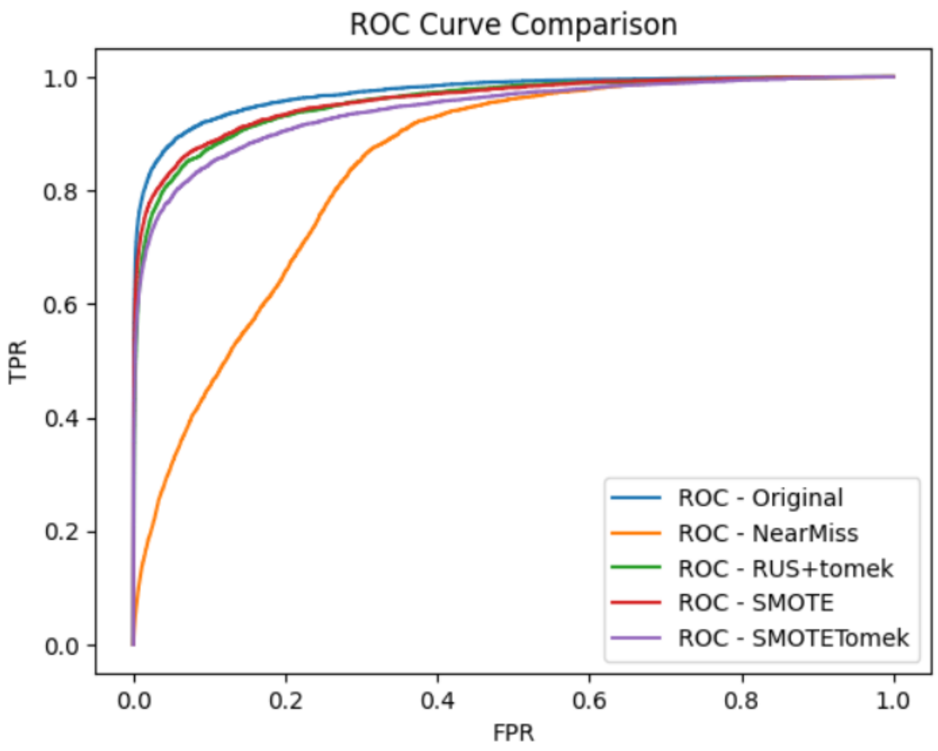

FIGURE 12 - ROC Curve Comparison

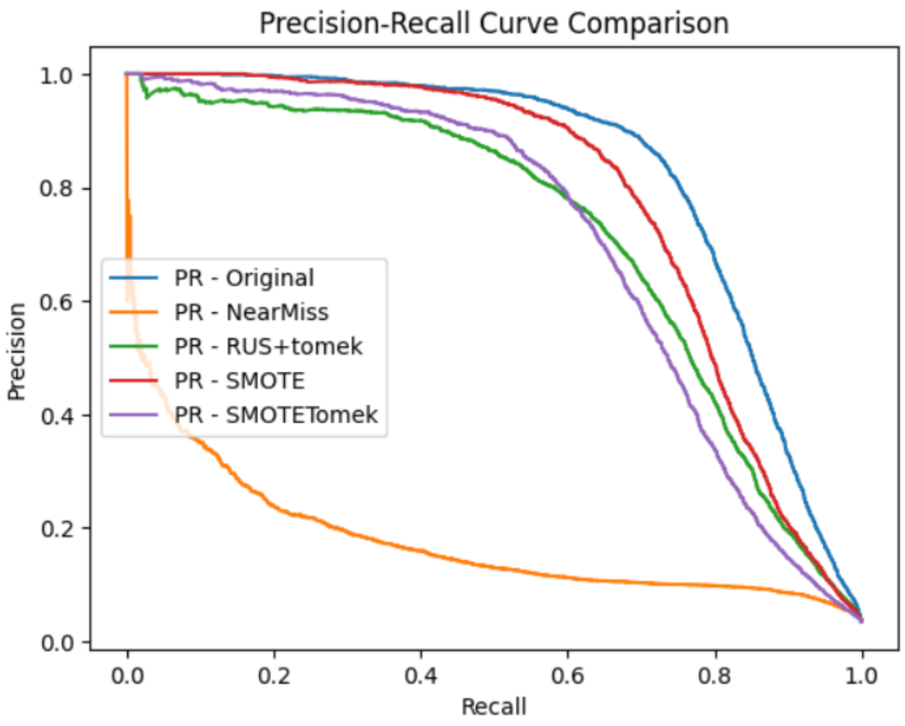

FIGURE 13 - PR Curve Comparison

As mentioned in Step 5 above, since SMOTETomek resamples on a small subset, its results are compared with SMOTE, which resamples on the same small subset. The comparison results are shown in Table 2 and Figures 14 to 20. Compared to SMOTE alone, SMOTETomek achieved a slightly better F1 score (0.632 vs 0.629) and AUC (0.942 vs 0.940), with marginal improvements in recall (0.486 vs 0.484) and precision (0.902 vs 0.900). This indicates a more balanced classifier that retains the precision strength of SMOTE while improving on its recall weakness by removing ambiguous examples from the majority class.

TABLE II

PERFORMANCE COMPARISON ON A SUBSET

	Accuracy	Precision	Recall	F1	AUC
SMOTE	0.98006062	0.90009000	0.48390999	0.62942564	0.94048453
SMOTETomek	0.98017069	0.90174966	0.48632954	0.63187676	0.94173635

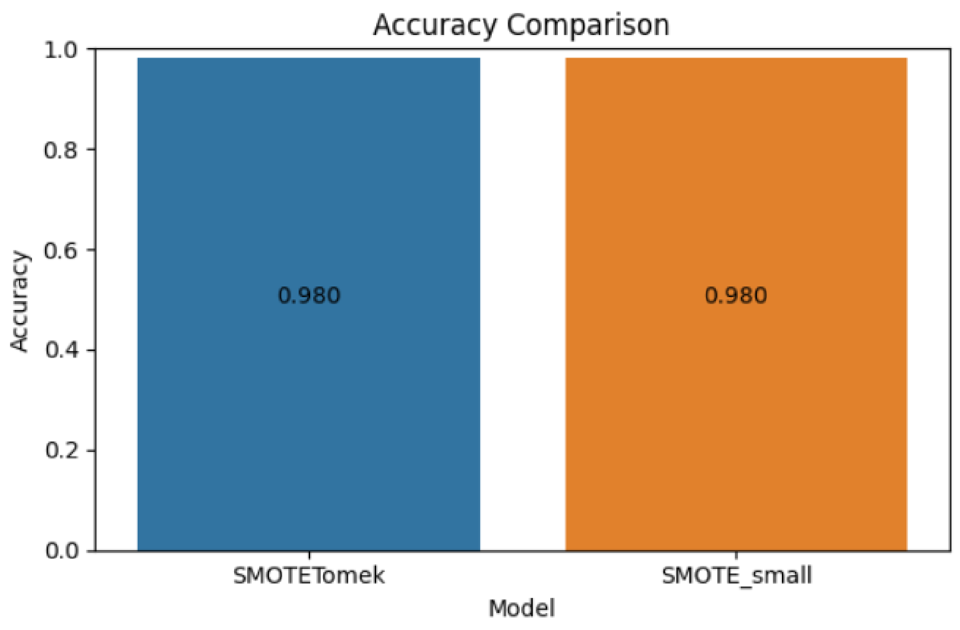

FIGURE 14 - Accuracy Comparison On Subset

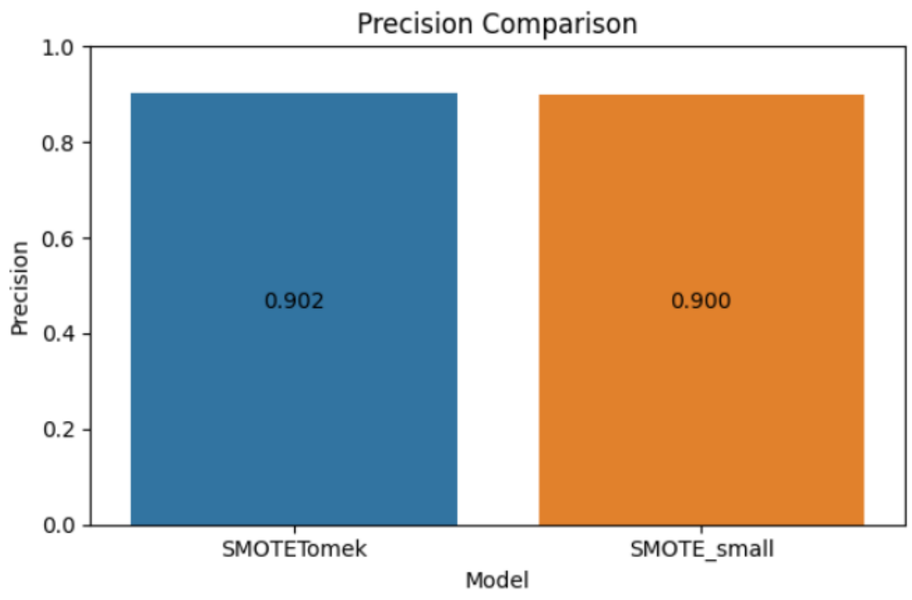

FIGURE 15 - Precision Comparison On Subset

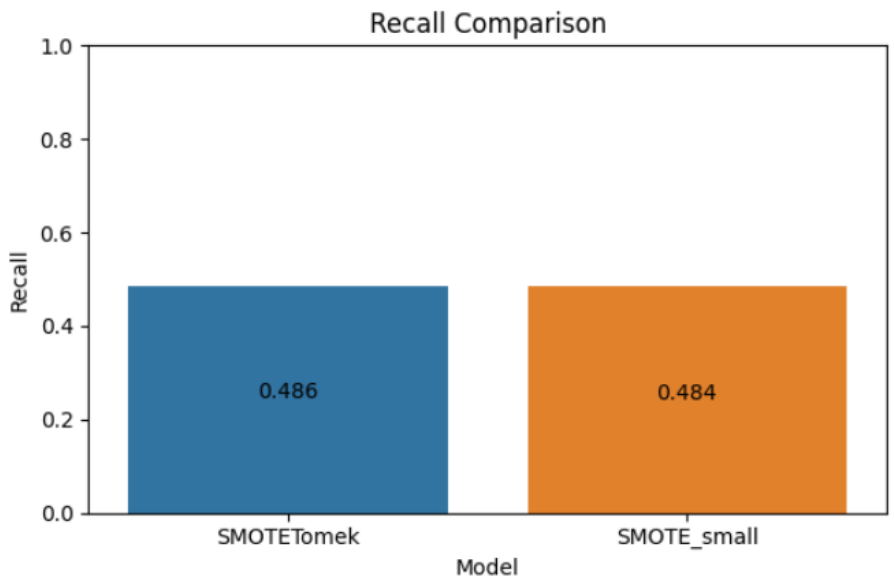

FIGURE 16 - Recall Comparison On Subset

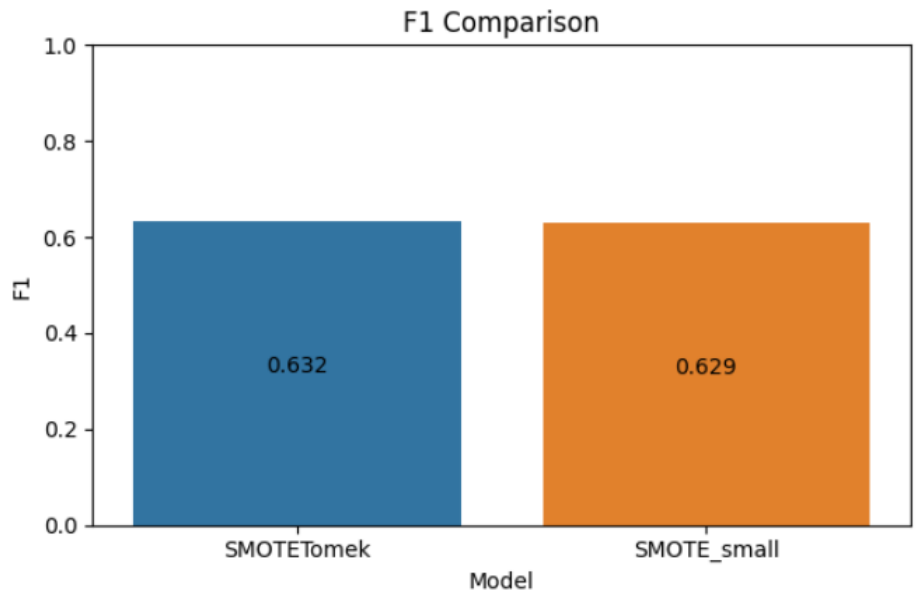

FIGURE 17 - F1 Comparison On Subset

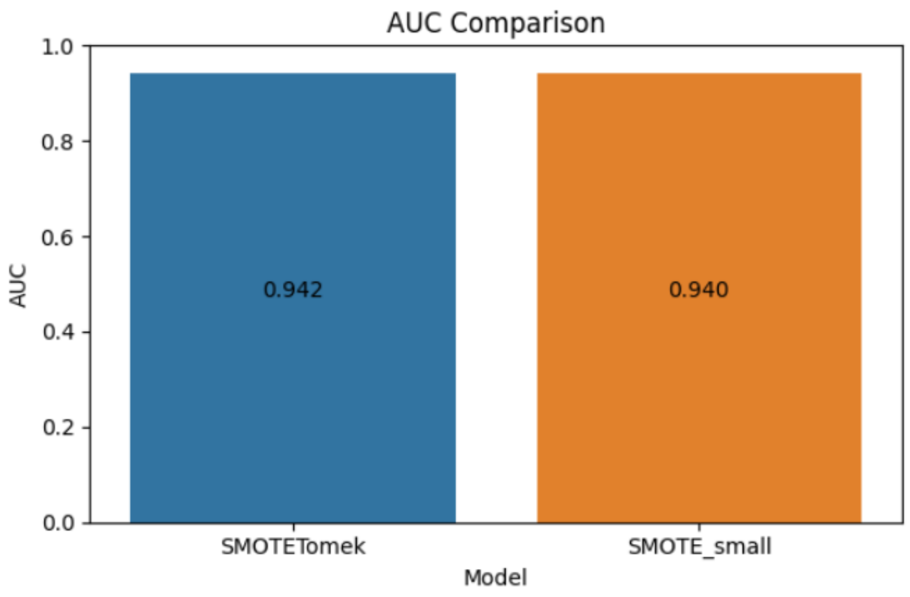

FIGURE 18 - AUC Comparison On Subset

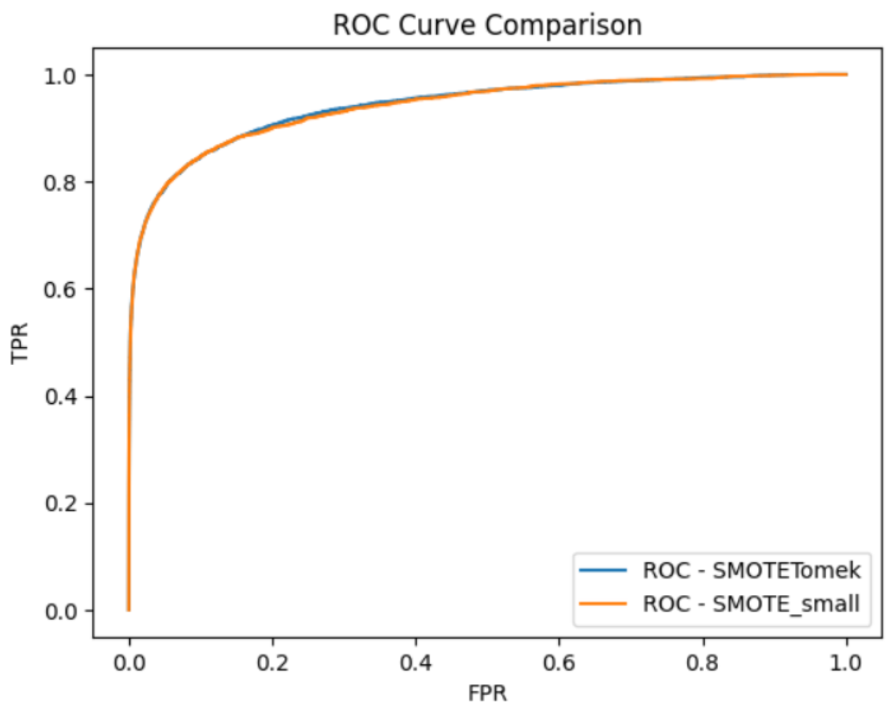

FIGURE 19 - ROC Comparison On Subset

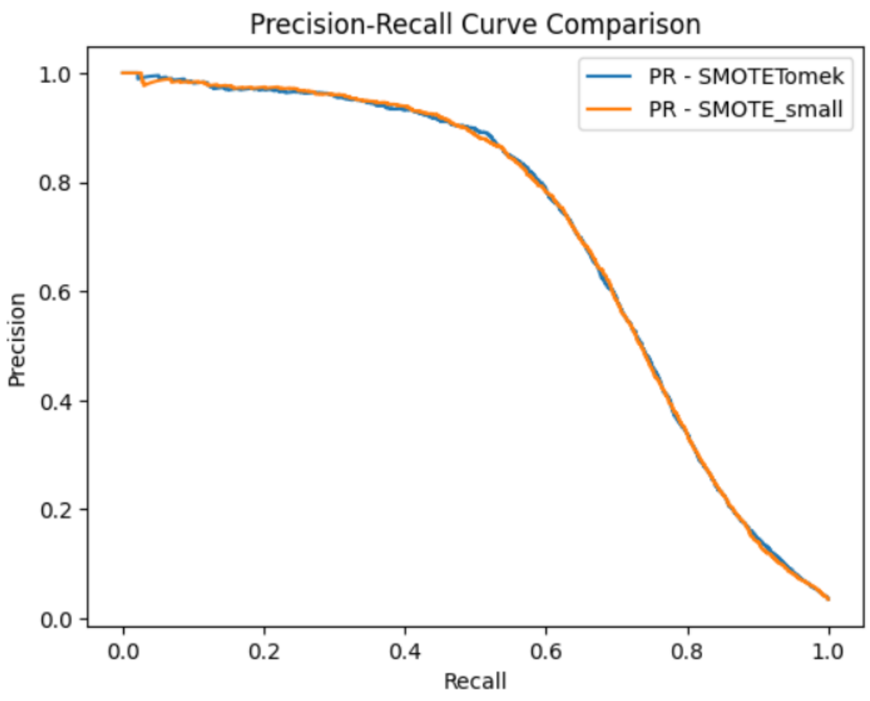

FIGURE 20 - PR Comparison On Subset

5. Discussion

This study aims to evaluate the effectiveness of different sampling techniques in dealing with class imbalance in fraud detection on the IEEE-CIS Fraud Detection dataset. The results show that the built-in scale_pos_weight parameter of XGBoost has shown strong performance, achieving the highest AUC (0.972) and a good balance between recall (0.806) and precision (0.654) for the model trained on the original (imbalanced) dataset. This suggests that algorithm-level adjustments may be effective enough for moderately imbalanced datasets and more effective than external resampling.

Among the resampling techniques, SMOTE and SMOTETomek achieved high precision (above 0.90), indicating that they are able to reduce false positives. However, this comes at the expense of a decrease in recall. In fraud detection, failing to capture actual fraudulent transactions usually results in greater financial losses than mislabeling legitimate transactions. Therefore, this trade-off may not be in line with actual business needs, especially in high-risk financial environments.

On the other hand, NearMiss achieved the highest recall (93.8%), meaning that it captured almost all fraud cases. However, its precision plummets to 7.5%, meaning that more than 90% of the flagged transactions are false positives. Such a model is not practical for real-time systems as it would impose a huge burden on human investigators and interfere with legitimate user activities. This again proves that aggressive undersampling, while improving recall, is not suitable for production environments where both accuracy and user experience are critical.

RUS + Tomek offers a more balanced compromise, improving recall (0.848) while maintaining reasonable precision (0.307). This suggests that intelligently removing noise or borderline samples can lead to competitive performance. However, it is still lower than the AUC achieved by the original model, which further supports the use of XGBoost's native class weighting feature.

5.1 Comparison with Previous Studies

To validate the results of this study, we compared the experimental results with those of previous researchers using the same IEEE-CIS Fraud Detection dataset.

A study by Nguyen et al. (2022) adopted SMOTE for oversampling and used CatBoost as the classifier. Their best performance was 0.9864 accuracy and 0.974 AUC. In comparison, our native XGBoost model achieved an accuracy of 0.978 and an AUC of 0.972. Although their model achieved marginally better scores, it required additional preprocessing via SMOTE and leveraged a more complex boosting framework that integrates categorical encoding internally.

A recent study by Najadat et al. (2020) adopted a deep neural architecture combining BiLSTM and BiGRU with max pooling layers. Impressive results were achieved on the same IEEE-CIS dataset with an AUC of 0.9137 and a recall of 0.9459 using random oversampling techniques. These results demonstrate the potential of complex neural network architectures in capturing temporal and sequential patterns in transaction data and improving fraud detection performance, especially in detecting more fraud cases.

Compared with the above studies that pursue extreme model performance, our XGBoost-based approach emphasizes model simplicity, faster training speed, and practical deployment capabilities, which is of practical value for the extensive comparison of different sampling strategies. At the same time, high AUC (0.972) and balanced recall (0.804) can be achieved without complex model structures.

5.2 Business Considerations

In real-world fraud detection, several key trade-offs must be considered. For example, in large-scale transaction screening, a model with high recall, like NearMiss, may be acceptable to ensure that no fraudulent transactions are missed. However, for real-time systems where user experience and system latency are critical, models with high accuracy and low false positive rates, such as native XGBoost models, are more appropriate.

Importantly, modern fraud detection systems operate under strict latency constraints and must be frequently retrained to accommodate concept drift (Dal Pozzolo et al., 2018). This makes fast, incremental training methods a more desirable choice. Sampling methods such as SMOTE or SMOTETomek, while effective offline, often incur significant computational overhead and are difficult to update easily in real-time processes.

Deep learning models such as BiLSTM-BiGRU require a lot of computing resources, longer training time, and are generally more difficult to interpret, which may pose a challenge for real-time deployment in high-throughput financial systems. In addition, compared with decision tree-based algorithms such as XGBoost, such models are less transparent. Transparency is also an important consideration in regulated industries such as finance.

Recent studies (Leevy et al., 2023) have shown that threshold optimization alone can outperform resampling in many cases. In practice, financial institutions often prefer model calibration, cost-sensitive learning, and threshold tuning over resampling, especially in production environments where interpretability, retraining speed, and real-time deployment are critical.

6. CONCLUSIONS

Based on the IEEE-CIS Fraud Detection dataset, this study explores the effectiveness of various resampling strategies in addressing the class imbalance problem in financial fraud detection. Based on the XGBoost algorithm as the base classifier, the study evaluates the performance of the original model and four resampling methods (NearMiss, RUS with Tomek Links, SMOTE, and SMOTETomek). The study uses a series of evaluation metrics - accuracy, precision, recall, F1 score, AUC, ROC curve, and PR curve - to fully evaluate the advantages and disadvantages of each method.

The results show that the original imbalanced dataset achieves the highest AUC (0.971) and high recall (0.804) when used with XGBoost's native scale_pos_weight parameter, indicating that algorithm-level adjustments can be very effective. SMOTE and SMOTETomek improve precision but reduce recall, indicating that this conservative prediction method may miss actual fraud cases. Although NearMiss achieves the highest recall, its precision and overall accuracy are too low to be practical for real-world applications.

These trade-offs are critical from a business perspective. In high-volume, real-time fraud detection systems, models must strike a balance between capturing fraudulent transactions (recall) and avoiding unnecessary noise (precision). The results show that resampling methods are not always superior to class weighting, especially in production environments where time is limited and retraining the model is required.

In summary, while resampling remains a valuable technique in imbalanced learning, practical deployment in financial fraud detection requires careful consideration of operational costs, model complexity, and business constraints. The native functionality of algorithms like XGBoost may already provide sufficient performance for many practical scenarios.

This study has several limitations. First, feature engineering is applied to the entire dataset before it is split into training and test sets, which may introduce data leakage and optimism bias. Second, due to hardware limitations, SMOTETomek was evaluated only on a reduced sample of 120k observations, which may not reflect its full potential on the entire dataset. Third, the fraud rate of the dataset used was 3.5%, which is imbalanced but not extreme. Results may differ for datasets with more severe imbalances (e.g., fraud rates < 1%).

In addition, models built using synthetic samples SMOTE, may face interpretability challenges because synthetic features may not correspond to real transactions. This may limit their practical deployment in systems that require interpretability and traceability, especially under regulatory scrutiny.

Future work could explore approaches such as cost-sensitive learning or online boosting, as well as threshold optimization as an alternative to resampling to handle more imbalanced datasets, etc.

References

Alarfaj, F. K., Malik, I., Khan, H. U., Almusallam, N., Ramzan, M., & Ahmed, M. (2022). Credit Card Fraud Detection Using State-of-the-Art Machine Learning and Deep Learning Algorithms. *IEEE Access*, *10*, 39700–39715. https://doi.org/10.1109/ACCESS.2022.3166891

Almazroi, A. A., & Ayub, N. (2023). Online Payment Fraud Detection Model Using Machine Learning Techniques. *IEEE Access*, *11*, 137188–137203. https://doi.org/10.1109/ACCESS.2023.3339226

Bin Sulaiman, R., Schetinin, V., & Sant, P. (2022). Review of Machine Learning Approach on Credit Card Fraud Detection. *Human-Centric Intelligent Systems*, *2*(1–2), 55–68. https://doi.org/10.1007/s44230-022-00004-0

Brandt, J., & Lanzén, E. (2020). *A Comparative Review of SMOTE and ADASYN in Imbalanced Data Classification* [Uppsala University]. https://www.diva-portal.org/smash/get/diva2:1519153/FULLTEXT01.pdf

Carvalho, M., Pinho, A. J., & Brás, S. (2025). Resampling approaches to handle class imbalance: A review from a data perspective. *Journal of Big Data*, *12*(1), 71. https://doi.org/10.1186/s40537-025-01119-4

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic Minority Over-sampling Technique. *Journal of Artificial Intelligence Research*, *16*, 321–357. https://doi.org/10.1613/jair.953

Cheah, P. C. Y., Yang, Y., & Lee, B. G. (2023). Enhancing Financial Fraud Detection through Addressing Class Imbalance Using Hybrid SMOTE-GAN Techniques. *International Journal of Financial Studies*, *11*(3), 110. https://doi.org/10.3390/ijfs11030110

Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 785–794. https://doi.org/10.1145/2939672.2939785

Costa, C. J., & Aparicio, J. T. (2020, June). POST-DS: A methodology to boost data science. In *2020 15th iberian conference on information systems and technologies (CISTI)* (pp. 1-6). IEEE. https://doi.org/10.23919/CISTI49556.2020.9140932

Costa, C.J., Aparicio, J.T. (2021). A Methodology to Boost Data Science in the Context of COVID-19. In: Arabnia, H.R., et al. *Advances in Parallel & Distributed Processing, and Applications. Transactions on Computational Science and Computational Intelligence*. Springer, Cham. https://doi.org/10.1007/978-3-030-69984-0_7

Dal Pozzolo, A., Boracchi, G., Caelen, O., Alippi, C., & Bontempi, G. (2018). Credit Card Fraud Detection: A Realistic Modeling and a Novel Learning Strategy. *IEEE Transactions on Neural Networks and Learning Systems*, *29*(8), 3784–3797. https://doi.org/10.1109/TNNLS.2017.2736643

de la Bourdonnaye, F., & Daniel, F. (2022). *Evaluating resampling methods on a real-life highly imbalanced online credit card payments dataset* (Version 1). arXiv. https://doi.org/10.48550/ARXIV.2206.13152

Ghosh, K., Bellinger, C., Corizzo, R., Branco, P., Krawczyk, B., & Japkowicz, N. (2024). The class imbalance problem in deep learning. *Machine Learning*, *113*(7), 4845–4901. https://doi.org/10.1007/s10994-022-06268-8

Hasanin, T., & Khoshgoftaar, T. (2018). The Effects of Random Undersampling with Simulated Class Imbalance for Big Data. *2018 IEEE International Conference on Information Reuse and Integration (IRI)*, 70–79. https://doi.org/10.1109/IRI.2018.00018

IEEE Computational Intelligence Society. (2019). *IEEE-CIS Fraud Detection*. Kaggle. https://kaggle.com/competitions/ieee-fraud-detection

Ileberi, E., Sun, Y., & Wang, Z. (2021). Performance Evaluation of Machine Learning Methods for Credit Card Fraud Detection Using SMOTE and AdaBoost. *IEEE Access*, *9*, 165286–165294. https://doi.org/10.1109/ACCESS.2021.3134330

Kalid, S. N., Khor, K.-C., Ng, K.-H., & Tong, G.-K. (2024). Detecting Frauds and Payment Defaults on Credit Card Data Inherited With Imbalanced Class Distribution and Overlapping Class Problems: A Systematic Review. *IEEE Access*, *12*, 23636–23652. https://doi.org/10.1109/ACCESS.2024.3362831

Khushi, M., Shaukat, K., Alam, T. M., Hameed, I. A., Uddin, S., Luo, S., Yang, X., & Reyes, M. C. (2021). A Comparative Performance Analysis of Data Resampling Methods on Imbalance Medical Data. *IEEE Access*, *9*, 109960–109975. https://doi.org/10.1109/ACCESS.2021.3102399

Leevy, J. L., Johnson, J. M., Hancock, J., & Khoshgoftaar, T. M. (2023). Threshold optimization and random undersampling for imbalanced credit card data. *Journal of Big Data*, *10*(1), 58. https://doi.org/10.1186/s40537-023-00738-z

Li, Zhu. (2025, June). *Credit-card-fraud-detection-resampling*. Github. https://github.com/lizh1994/credit-card-fraud-detection-resampling

Meng, C., Zhou, L., & Liu, B. (2020). A Case Study in Credit Fraud Detection With SMOTE and XGBoost. *Journal of Physics: Conference Series*, *1601*(5), 052016. https://doi.org/10.1088/1742-6596/1601/5/052016

Mittal, P., Lallie, H. S., & Titis, E. (2025). Impact of imbalanced datasets on ML algorithms for malware classification. *Information Security Journal: A Global Perspective*, *34*(3), 251–264. https://doi.org/10.1080/19393555.2025.2459736

Mohammed, R., Rawashdeh, J., & Abdullah, M. (2020). Machine Learning with Oversampling and Undersampling Techniques: Overview Study and Experimental Results. *2020 11th International Conference on Information and Communication Systems (ICICS)*, 243–248. https://doi.org/10.1109/ICICS49469.2020.239556

Mqadi, N. M., Naicker, N., & Adeliyi, T. (2021). Solving Misclassification of the Credit Card Imbalance Problem Using Near Miss. *Mathematical Problems in Engineering*, *2021*, 1–16. https://doi.org/10.1155/2021/7194728

Muaz, A., Jayabalan, M., & Thiruchelvam, V. (2020). A Comparison of Data Sampling Techniques for Credit Card Fraud Detection. *International Journal of Advanced Computer Science and Applications*, *11*(6). https://doi.org/10.14569/IJACSA.2020.0110660

Najadat, H., Altiti, O., Aqouleh, A. A., & Younes, M. (2020). Credit Card Fraud Detection Based on Machine and Deep Learning. *2020 11th International Conference on Information and Communication Systems (ICICS)*, 204–208. https://doi.org/10.1109/ICICS49469.2020.239524

Nguyen, N., Duong, T., Chau, T., Nguyen, V.-H., Trinh, T., Tran, D., & Ho, T. (2022). A Proposed Model for Card Fraud Detection Based on CatBoost and Deep Neural Network. *IEEE Access*, *10*, 96852–96861. https://doi.org/10.1109/ACCESS.2022.3205416

Prusa, J., Khoshgoftaar, T. M., Dittman, D. J., & Napolitano, A. (2015). Using Random Undersampling to Alleviate Class Imbalance on Tweet Sentiment Data. *2015 IEEE International Conference on Information Reuse and Integration*, 197–202. https://doi.org/10.1109/IRI.2015.39

Rubaidi, Z. S., Ammar, B. B., & Aouicha, M. B. (2022). Fraud Detection Using Large-scale Imbalance Dataset. *International Journal on Artificial Intelligence Tools*, *31*(08), 2250037. https://doi.org/10.1142/S0218213022500373

Shabrina Assyifa, D., & Luthfiarta, A. (2024). SMOTE-Tomek Re-sampling Based on Random Forest Method to Overcome Unbalanced Data for Multi-class Classification. *Inform : Jurnal Ilmiah Bidang Teknologi Informasi Dan Komunikasi*, *9*(2), 151–160. https://doi.org/10.25139/inform.v9i2.8410

Singh, A., Ranjan, R. K., & Tiwari, A. (2022). Credit Card Fraud Detection under Extreme Imbalanced Data: A Comparative Study of Data-level Algorithms. *Journal of Experimental & Theoretical Artificial Intelligence*, *34*(4), 571–598. https://doi.org/10.1080/0952813X.2021.1907795

Tarawneh, A. S., Hassanat, A. B., Altarawneh, G. A., & Almuhaimeed, A. (2022). Stop Oversampling for Class Imbalance Learning: A Review. *IEEE Access*, *10*, 47643–47660. https://doi.org/10.1109/ACCESS.2022.3169512

Tiwari, P., Mehta, S., Sakhuja, N., Kumar, J., & Singh, A. K. (2021). *Credit Card Fraud Detection using Machine Learning: A Study* (No. arXiv:2108.10005). arXiv. https://doi.org/10.48550/arXiv.2108.10005

Trisanto, D., Rismawati, N, Mulya, M.F. & Kurniadi, F. I. (2021). Modified Focal Loss in Imbalanced XGBoost for Credit Card Fraud Detection. *International Journal of Intelligent Engineering and Systems*, *14*(4), 350–358. https://doi.org/10.22266/ijies2021.0831.31

Wang, Z., Wu, C., Zheng, K., Niu, X., & Wang, X. (2019). SMOTETomek-Based Resampling for Personality Recognition. *IEEE Access*, *7*, 129678–129689. https://doi.org/10.1109/ACCESS.2019.2940061

Zhang, Y., Tong, J., Wang, Z., & Gao, F. (2020). Customer Transaction Fraud Detection Using Xgboost Model. *2020 International Conference on Computer Engineering and Application (ICCEA)*, 554–558. https://doi.org/10.1109/ICCEA50009.2020.00122