Credit Card Fraud Detection Using Machine Learning

Group 11: Arnav Hiray, Haoming Shi, Jiafeng Song, Zihan Xu, Yijia Zheng

Abstract

Credit card fraud detection is a critical concern for financial institutions and individuals due to its significant financial impact and potential harm to personal credit. Machine learning has emerged as a promising solution for credit card fraud detection, leveraging its ability to analyze large datasets and identify complex patterns and anomalies. This paper presents a comprehensive study on the application of machine learning algorithms for credit card fraud detection, including data preprocessing, model selection, and evaluation. Several machine learning algorithms, such as logistic regression, Gaussian Naive Bayes, decision trees, random forests, and XGBoost, are evaluated on a real-world dataset comprising historical credit card transactions. The results demonstrate the effectiveness of machine learning algorithms in detecting credit card fraud, with variations in performance across different algorithms, and provide insights into the impact of data preprocessing techniques and the performance of different models. By leveraging the power of machine learning, financial institutions can mitigate financial losses, protect their customers, and maintain trust in the credit card industry.

1 Introduction

Credit card fraud poses a significant threat to both financial institutions and individuals, resulting in substantial financial losses and potential damage to personal credit. Traditional methods of fraud detection, such as rule-based systems and manual review, have shown limitations in effectively detecting and preventing increasingly sophisticated fraud techniques. As fraudsters constantly adapt their strategies, there is a critical need for more robust and efficient approaches to combat credit card fraud.

Machine learning has emerged as a promising solution for credit card fraud detection, leveraging its ability to analyze vast amounts of data and identify complex patterns and anomalies. By utilizing advanced algorithms, machine learning models can learn from historical transaction data and accurately classify transactions as either fraudulent or legitimate. This offers the potential to detect fraud in real-time, significantly reducing financial losses and protecting both financial institutions and cardholders.

This paper aims to conduct a comparative study of different machine learning algorithms for credit card fraud detection. We aim to evaluate the performance and effectiveness of various algorithms, including logistic regression, Gaussian Naive Bayes, decision trees, random forests, and XGBoost. By comparing these algorithms, we can gain insights into their strengths and weaknesses in accurately detecting credit card fraud.

The study also addresses critical aspects of data preprocessing, including data cleaning and handling the class imbalance issue. Since fraudulent transactions are often rare compared to legitimate ones, the dataset is imbalanced, which can lead to biased models that favor the majority class. Techniques such as oversampling the minority class, undersampling the majority class, or utilizing synthetic data generation methods like SMOTE (Synthetic Minority Over-sampling Technique) can help address this issue and improve the model's performance on detecting fraudulent transactions.

Additionally, we explore the impact of different evaluation metrics and techniques to assess the performance of the models, such as precision, recall, F1-score, and receiver operating characteristic (ROC) curves.

Overall, this paper aims to demonstrate the potential of machine learning in enhancing credit card fraud detection, paving the way for more robust and proactive fraud prevention measures. By leveraging the power of machine learning, financial institutions can minimize losses, protect their customers, and maintain trust in the credit card industry.

2 Data Preprocessing

2.1 Dataset Overview

We utilized a dataset from Kaggle[1], comprising 2-day credit card transactions made by European cardholders in September 2013.

The dataset consists of numerical input variables derived from a PCA transformation. Due to confidentiality constraints, we cannot access the original features or additional background information. Principal components V1 to V28 are the results of the PCA, while the 'Time' and 'Amount' features remain untransformed. The 'Time' feature represents the seconds elapsed between each transaction and the first one in the dataset, while the 'Amount' feature indicates the transaction amount.The 'Class' feature serves as the response variable, with 1 indicating fraud and 0 otherwise.

Upon removing duplicate data and examining missing values, we obtained a dataset with 473 fraudulent transactions out of 283,726 total transactions. The dataset is highly imbalanced, with fraud cases constituting only 0.167 % of all transactions.

2.2 Resampling Methods

In our study, we addressed the issue of class imbalance at the data level by employing a variety of resampling techniques. These approaches generally fall into one of the three main categories: oversampling, undersampling, and hybrid methods. Figure 1[2]demonstrates how random undersampling and random oversampling work.

To prevent data leakage and ensure a reliable assessment of our model's performance, we employed a stratified splitting strategy. This involved dividing the dataset into train and test sets, with an 80/20 split ratio. By adopting this method, we were able to maintain the same class distribution in both subsets before implementing any resampling techniques. Table 1 presents the sizes of the train and test datasets after implementing various resampling methods.

Table 1: Sizes of Resampled Datasets

Methods	Train Size	Test Size
Original Dataset	226980	56746
Random Undersampling	756	56746
Random Oversampling	453204	56746
SMOTE	453204	56746
ADASYN	453269	56746
SMOTE-Tomek	452010	56746
SMOTE-ENN	427083	56746

[1]Source of dataset: https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud

[2]Source of illustration for the comparison between random undersampling and random oversampling: https://www.kaggle.com/code/rafjaa/resampling-strategies-for-imbalanced-datasets/notebook#t7

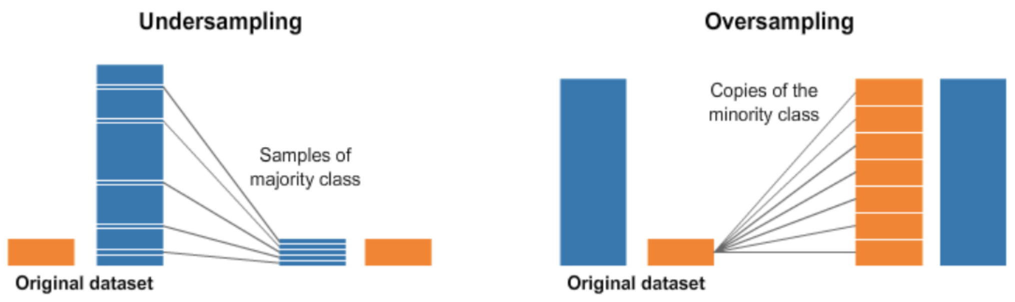


Figure 1: Illustration of random undersampling and random oversampling

2.2.1 Undersampling

We employed three undersampling techniques to reduce the number of majority target instances and achieve a 50:50 class imbalance ratio.

The simplest approach is random undersampling, which attempts to balance target distributions by randomly removing instances from the majority class. The major drawback of this method is that it may eliminate potentially valuable data crucial for building effective classifier models(Batista et al., 2004). However, random undersampling can be useful when dealing with large datasets(Mohammed et al., 2020).

We also explored more advanced strategies aimed at removing samples from overlapping regions to create clearer decision boundaries. Specifically, we implemented Tomek Links and Edited Nearest-Neighbors (ENN) techniques. Both methods leverage the nearest-neighbor rule for sample removal. As illustrated in Figure 2[3], a Tomek Link(Tomek, 1976) is defined as a pair of instances belonging to different classes, where each instance is the nearest neighbor of the other.The Tomek Links technique aims to create a more distinct separation between classes by removing majority target instances that form such links. On the other hand, the Edited Nearest-Neighbors (ENN)(Wilson, 1972) technique focuses on identifying and removing instances that are misclassified based on their nearest neighbors' class labels. For a given instance, if its class label is different from the majority of its k nearest neighbors' class labels (typically, k=3), it is considered misclassified and subsequently removed from the dataset.

Nevertheless, in practice, both Tomek Links and ENN may be less effective in addressing large class imbalances, as they only remove a relatively small number of instances from the majority class. In our case, Tomek Links and ENN eliminated 442 and 740 majority instances, respectively, while the proportion of fraud samples still remained less than 0.167 %. Consequently, we decided not to proceed with these two methods.

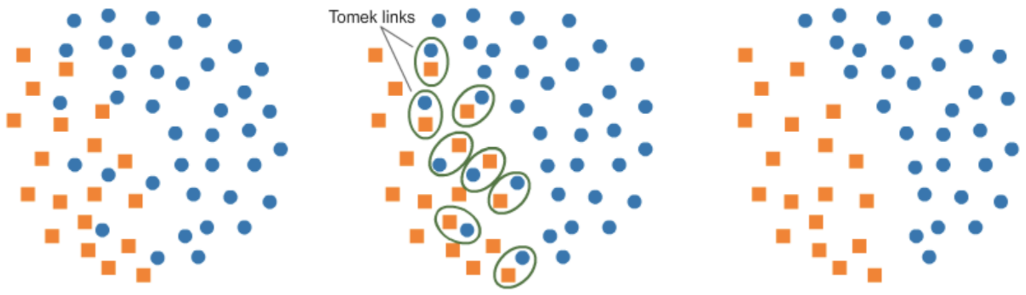


Figure 2: Illustration of Tomek Links

[3]Source of illustration for Tomek Links: https://www.kaggle.com/code/rafjaa/resampling-strategies-for-imbalanced-datasets/notebook#t7

2.2.2 Oversampling

Our project adopted three oversampling methods to balance class distribution by generating new samples for the minority class. It is important to note that oversampling increases dataset size, resulting in longer training times.

We first used random oversampling, which simply duplicates existing minority samples. However, prior research indicates that this can lead to overfitting, as it replicates existing samples without introducing new information(Mohammed et al., 2020).

To address this limitation, we experimented with two advanced techniques, SMOTE and ADASYN, both of which generate synthetic minority samples as shown in Figure 3[4]. SMOTE (Synthetic Minority Oversampling Technique)(Chawla et al., 2002) creates synthetic instances by interpolating between existing samples and their k-nearest neighbors in the same class.ADASYN(Adaptive Synthetic Sampling) (He et al., 2008) extends SMOTE by adaptively generating instances based on the minority class's local distribution, producing more synthetic data for harader-to-learn observations compared to SMOTE's uniform sampling strategy (Brandt & Lanzén, 2021).

Previous studies have shown that SMOTE may not accurately reproduce the original sample distribution, potentially affecting classifier performance(Zheng et al., 2015). Additionally, both methods can generate noisy samples that overlap with majority class instances Barua et al. (2011).

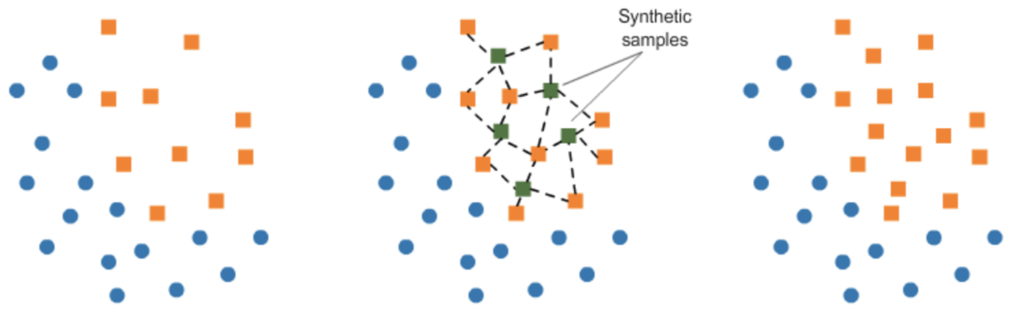


Figure 3: Illustration of generating synthetic samples

2.2.3 Hybrid Methods

Hybrid resampling methods combine the two previous techniques, which generate new samples for the minority class while using undersampling techniques to remove noisy samples. By integrating the strengths of both oversampling and undersampling, hybrid methods can potentially lead to better class separation and reduced overfitting[1].In our project, we utilized two popular hybrid methods—SMOTE-Tomek and SMOTE-ENN, with the latter generally removing more samples than the former.

3 Model Selection and Implementation

3.1 Classification Models

In order to evaluate the various resampling methods and their efficacies, we use five models that are appropriate for supervised binary classification: Logistic Regression, Gaussian Naive Bayes, Decision Tree, Random Forest, and XGBoost. Each model assigns one of two classes to each transaction: 0 or 1, representing a non-fraudulent or fraudulent transaction respectively. The output of each model is the probability that the given transaction belongs to either class, and the label with the higher probability is assigned.

[4]Source of illustration for generating synthetic samples: https://www.kaggle.com/code/rafjaa/resampling-strategies-for-imbalanced-datasets/notebook#t7

3.1.1 Linear Classifiers

There are two linear models we implement: Logistic Regression and Gaussian Naive Bayes.

Logistic Regression is a generalized linear model widely used in the financial industry, specifically pertaining to credit risk and financial risk management. Simplicity of the algorithm allows for efficient training of the model on large datasets as well as high interpretability. Logistic Regression is often used as a benchmark.

Gaussian Naive Bayes is a generative classification model that assumes the underlying features are independent and follow a gaussian distribution. Alike Logistic Regression, Gaussian Naive Bayes is also used in modeling credit risk. The "Naive" assumption of independent features can also make Gaussian Naive Bayes a good fit for high-dimensional data.

3.1.2 Tree-Based Classifiers

There are three tree-based classifiers which we explore.

The most basic of them, Decision Tree is a classification model that is non-parametric and generates a set of rules based on the input features for the purpose of classifying transactions. This tree-based process can often be visualized and hence provide a level of interpretability. It can also handle both categorical and numerical data. Decision Tree can be used as a benchmark for tree-based classification models.

Random Forest is a ensemble classification model that fits multiple Decision Tree classifiers on subsets of the dataset and features. Ensembling the prediction of multiple decision trees not only provides more stable predictions but also provides a more robust model than a single Decision Tree Classifier. Such a bagging method attempts to improve the model performance and reduce the risk of overfitting.

XGBoost is a powerful gradient boosting-based model that incorporates features such as regularization to handle overfitting and weighted quantile sketch for identifying optimal splits. This modifications along with the underlying gradient boosting-based algorithm attempts to minimize bias and underfitting.

3.2 Hyperparameter Tuning

Hyperparameter tuning is a process to determine the optimal values for a model's hyperparameters. We tune the hyperparameters using a validation set. The validation set is taken from the training data prior to model fitting, allowing for testing of the model's performance on unseen data. We use an exhaustive search (also known as Grid Search).

3.2.1 Scoring

Determining the correct scoring metric for hyperparameter tuning heavily depends on the dataset and domain knowledge. This research deals with a highly imbalanced credit card transaction dataset. As such, metrics that do not take into account label imbalance are not appropriate to use both for scoring the hyperparameter tuning and later on for evaluating resampling methods and models. As an example, observe a simple metric: accuracy. Given predicted labels $\hat{y}_i \in \hat{y}$ and the true labels $y_i \in y$, the accuracy can be measured as

$$\text{accuracy}(y, \hat{y}) = \frac{1}{n_{\text{samples}}} \sum_{i=0}^{n_{\text{samples}}-1} \mathbb{1}(y_i = \hat{y}_i)$$

While accuracy in many settings is a good measure of model performance, note that in an imbalanced setting such a metric will give a preference to the majority class. Thus when hyperparmeter tuning given accuracy as a scoring metric, a model which simply predicts the majority class will still have a high accuracy and can be chosen. However clearly such a model is inapropriate in this context. Metrics that weight both the majority and minority class scoring performance equally are often referred to as "macro-averaged".

The F1-Score is a widely used metric both for evaluating model performance and for hyperparameter tuning. The F1-Score can be calculated as

$$\text{F1-Score} = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

Taking into account for the class imbalance, our model chooses the best hyperparameter setting using the Macro F-1 Score defined as

$$\text{Macro F1-Score} = \frac{\text{F1-Score}_0 + \text{F1-Score}_1}{2}$$

The Macro F1-Score calculates the F1 score individually then averages the score. This is used in multi-class classification but can also be applied in binary classification.

4 Evaluation and Results

Binary classification models are trained and hyperparameters for models are optimized based on balanced F1 scores. After model training, due to the original dataset imbalance, we evaluated our model training performances utilizing precision-recall curves (PRC). To quantitatively compare the evaluation metrics of model performances, we calculated the area under precision-recall curves (AUPRC) and F1 scores of each model against all sampling methods.

4.1 Model training performances evaluation

With trained models, PRC for the original dataset is displayed in Figure 4. Both ensemble tree-based classification models Random Forest and XGBoost have the best performance on the unchanged train data. Logistic Regression and Decision Tree has lower AUPRC, which has an average performance of distinguishing the positive and negative classes. However, among all classifiers, Gaussian Naive Bayes has the least performance of failure to classify classes accurately.

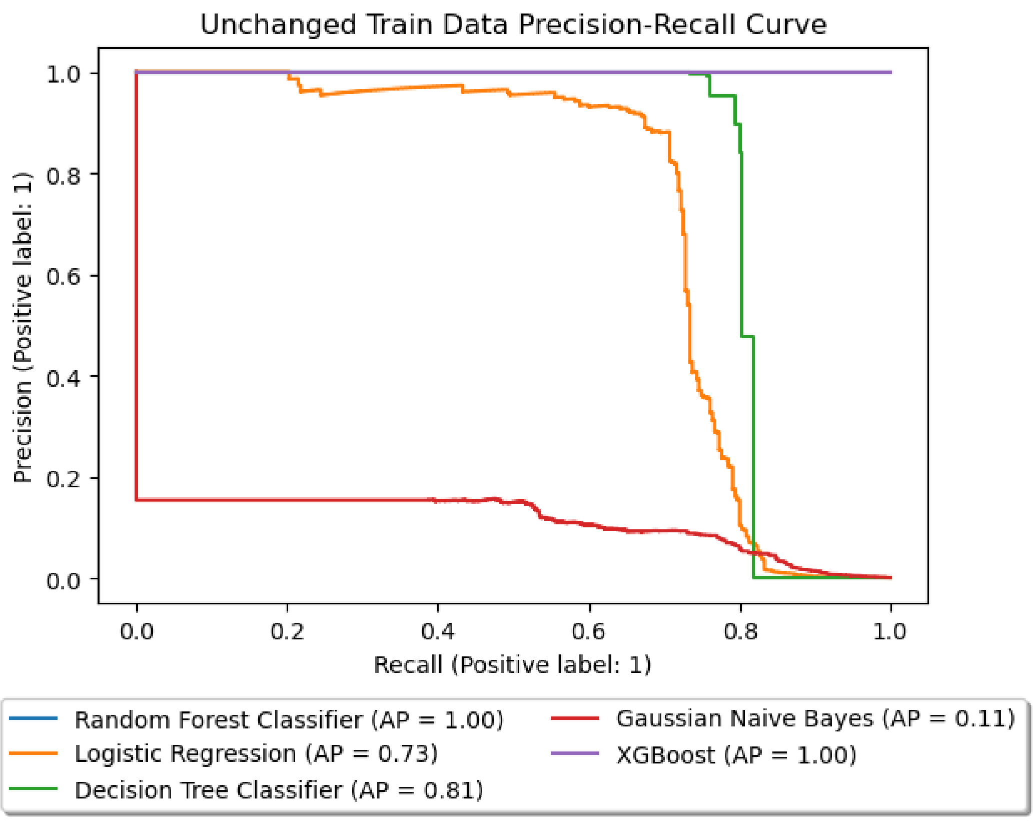
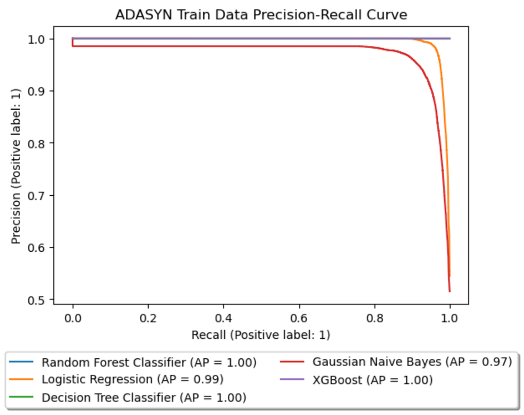
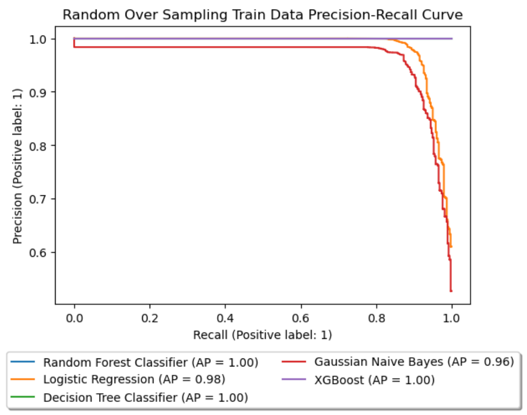


Figure 4: PRC for Unchanged Train Data

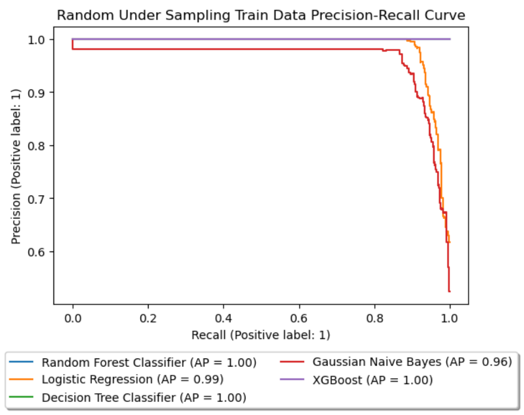
For balanced datasets in Figure 5, all trained models have considerably high AUPRC (>0.95) with all balanced datasets. This indicates that during training, selected classification models display no significant differences in precision and recall for both majority and minority classes for all balancing dataset sampling methods.



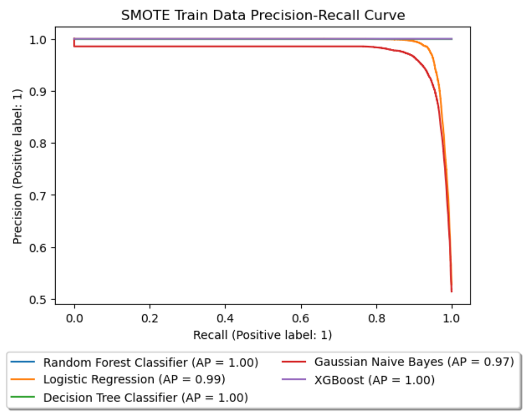
(a) PRC for ADASYN Train Data



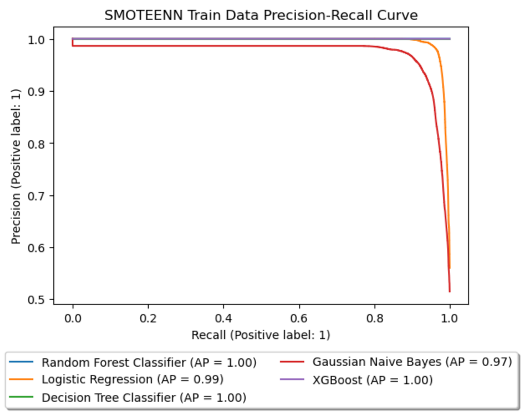
(b) PRC for Random Over Sampling Train Data



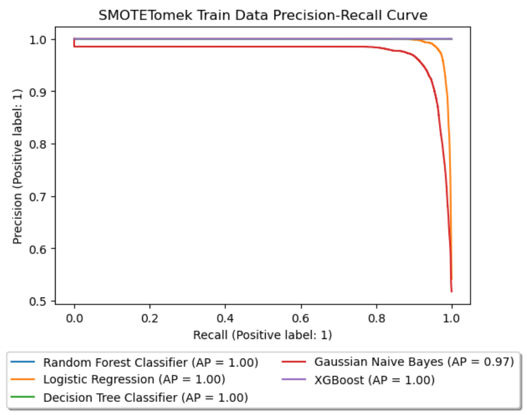
(c) PRC for Random Under Sampling Train Data



(d) PRC for SMOTE Train Data



(e) PRC for SMOTEENN Train Data



(f) PRC for SMOTETOMEK Train Data

Figure 5: PRC for balanced training datasets

4.2 Model testing performances evaluation

With trained models on the testing dataset, PRC for the unchanged test dataset is displayed in Figure 6. Both ensemble tree-based classification models Random Forest and XGBoost performances are consistent with unchanged train data. Both classification models have extremely high AUPRC value (>0.9) Not surprisingly, Logistic Regression and Decision Tree have lower AUPRC (close to 0.5) again which means an average performance of distinguishing the positive and negative classes while Gaussian Naive Bayes completed the classification task poorly on the unbalanced testing dataset. In addition, for testing on the unbalanced dataset, tree-based non-linear classification models (Random Forest, XGBoost, and Decision Tree) have outperformed linear classifiers (Gaussian Naive Bayes and Logistic Regression).

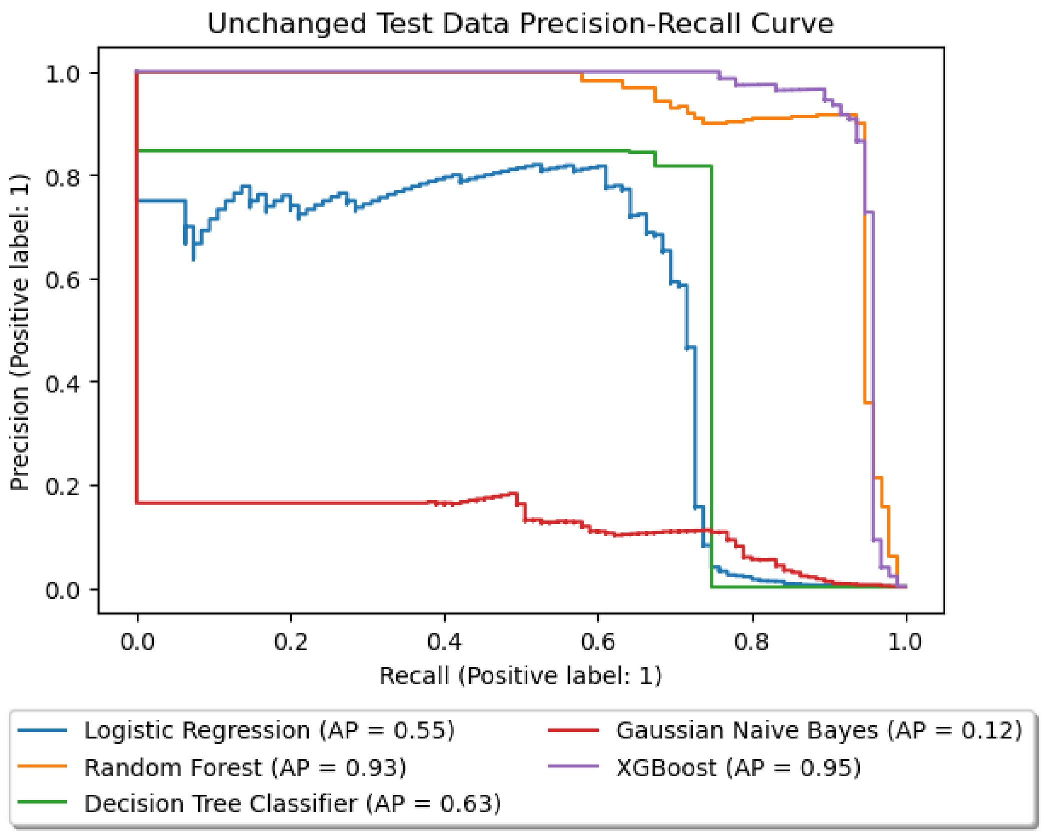
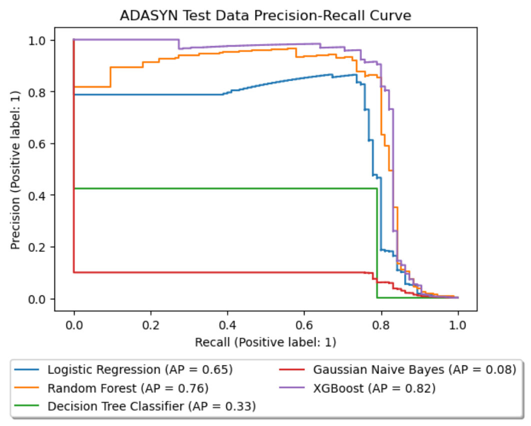
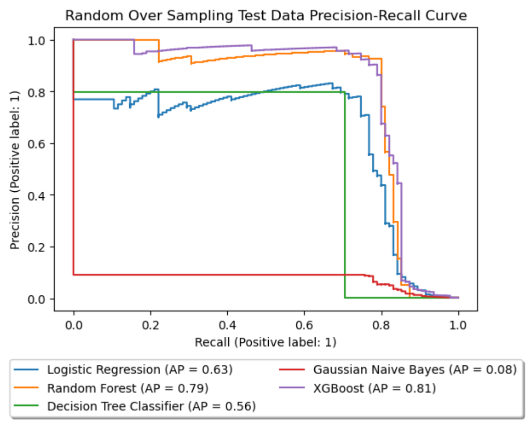


Figure 6: PRC for Unchanged Train Data

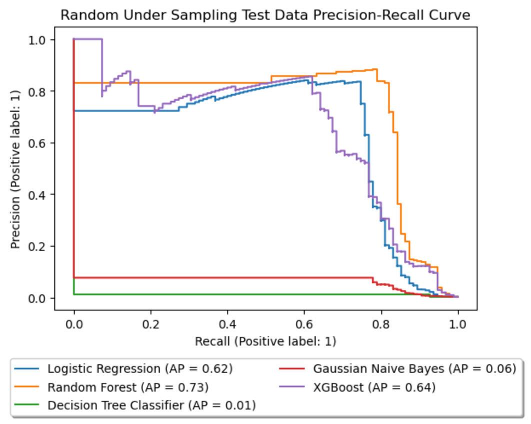
In Figure 7, SMOTEEEN sampling method has been shown to have the highest AUPRC value for almost all classification models comparing to other classifications, in which XGBoost classifier has achieved the highest AUPRC value (0.96) overall. For all balancing sampling methods, Random Forest and XGBoost have consistent high AUPRC (>0.9), which underlined that tree-based non-linear classifier has better performance on fraud detection task.



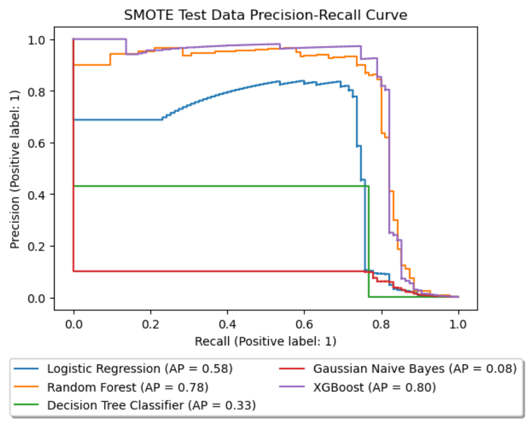
(a) PRC for ADASYN Train Data



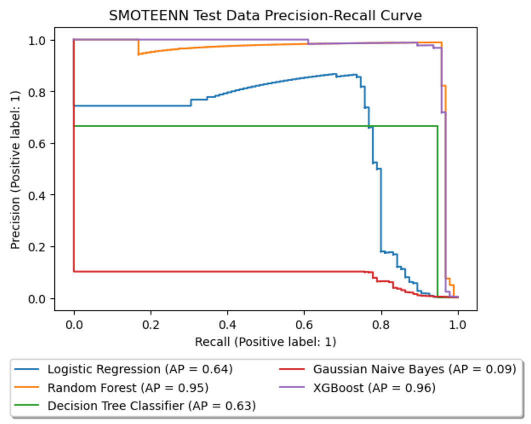
(b) PRC for Random Over Sampling Train Data



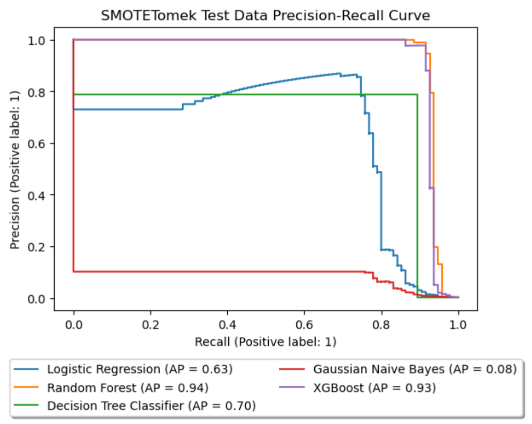
(c) PRC for Random Under Sampling Train Data



(d) PRC for SMOTE Train Data



(e) PRC for SMOTEENN Train Data



(f) PRC for SMOTETOMEK Train Data

Figure 7: PRC for balanced testing datasets

The average F1 Score for all balancing sampling methods is shown in Table 2. Logistic Regression and Gaussian Naive Bayes classifiers have comparatively low F1 scores from other classification models in all sampling methods. By looking at only F1 score, the majority of model performances do not have significant differences in different sampling methods compared to the original unbalanced dataset. However, the random forests model has slightly higher F1 scores in oversampling methods with SMOTE than the original unbalanced datasets. The random forest model with SMOTEENN has the highest F1 score (0.98).

Table 2: Average Macro F1 Score

	LR	Gaussian NB	XGB	Decision Tree	Random Forest
Original Dataset	0.72	0.58	0.96	0.87	0.96
Random Undersampling	0.52	0.53	0.54	0.48	0.55
Random Oversampling	0.53	0.54	0.92	0.87	0.91
SMOTE	0.53	0.55	0.91	0.78	0.91
ADASYN	0.56	0.55	0.91	0.78	0.91
SMOTE-Tomek	0.56	0.55	0.96	0.92	0.97
SMOTE-ENN	0.56	0.55	0.96	0.89	0.98

The average F1 Score for all balancing sampling methods is shown in Table 3. With a higher weight applied on precision, which is a better evaluation metric for the risk-sensitive models, all models for all sampling methods have higher F1 scores compared to F1 scores. XGBoost, Decision Tree, and Random Forest have higher F2 scores in SMOTE-Tomek and SMOTE-ENN compared to original unbalanced dataset and especially to other datasets.

Table 3: Average Macro F2 Score

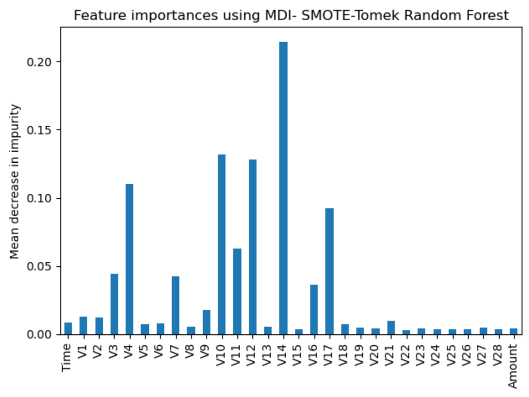
	LR	Gaussian NB	XGB	Decision Tree	Random Forest
Original Dataset	0.79	0.65	0.97	.85	.97
Random Undersampling	0.56	0.58	0.58	.47	0.61
Random Oversampling	0.57	0.59	0.90	0.86	0.88
SMOTE	0.58	0.60	0.90	.83	.90
ADASYN	0.62	0.60	0.91	0.84	0.90
SMOTE-Tomek	0.62	0.60	0.96	0.94	0.96
SMOTE-ENN	0.62	0.60	0.96	0.94	0.96

Using the Macro F2 Score instead of F1 is a crucial distinction: an undetected fraudulent transaction will have more negative effect than a legitimate transaction marked as fraudulent.

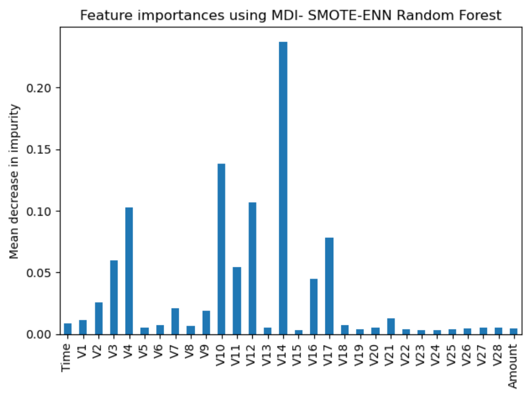
4.3 The Importance of Amount

Often a heuristic within the banking industry that is used to identify fraudulent transactions is flagging large transactions. However, do these models give importance to the transaction amount as well?

In order to quantify this, we take a further look into the Random Forest model on both the SMOTE-Tomek and SMOTE-ENN datasets as they had the best model performance. We can quantify feature importance using the Mean Decrease in Impurity (also known as Gini Importance). This calculates the number of times a particular feature is used to split a node in a tree weighted by the number of samples it splits. Thus a higher Mean Decrease in Impurity (MDI) would indicate higher feature importance.



(a) Mean Decrease in Impurity: SMOTE-Tomek



(b) Mean Decrease in Impurity: SMOTE-ENN

As one can see, Amount has a very low score. This indicates that the amount of a transaction may not be a large factor for detecting fraudulent activity. However, one thing to observe is that a few features, namely V4,V10,V2,V14 and V17 have a very large MDI and thus feature importance, particular V14. Due to the sensitive nature of the underlying data, we are unable to see the true features. However V1 corresponds to the first principal component and so on. Seeing features such as V10,V14, and V17 rank high in feature importance despite being the 10th, 14th, and 17th, principal direction indicates that reducing dimensions by removing the higher direction components would result in substantial loss to model performance in evaluating fraudulent transactions.

5 Conclusion and Future Work

Traditional credit fraud detection methods have been used for many years, but they have a number of limitations, such as not being able to detect fraud quickly enough, using low-quality data, being expensive to implement and maintain, and being susceptible to human error. Machine learning techniques can help to overcome these limitations by identifying fraud more quickly, accurately, and cheaply. This study investigated the use of machine learning techniques to detect credit fraud from two-day transactions made by European cardholders in September 2013. The study compared five different machine learning classifiers (logistic regression (LR), decision tree (DT), Gaussian naive Bayes (GNB), random forest (RF), and extreme gradient boosting (XGBoost)) and six different resampling methods (adaptive synthetic sampling (ADASYN), random over sampling, random under sampling, synthetic minority oversampling technique (SMOTE), SMOTE edited nearest neighbor (SMOTEENN), and SMOTE Tomek links (SMOTETOMEK)). The hyperparameters of each model were tuned using RandomizedSearchCV.

The study found that the random forest (RF) and extreme gradient boosting (XGBoost) classifiers, when combined with SMOTEENN hybrid resampling, performed best in detecting fraudulent transactions. Both algorithms were able to achieve a precision of over 0.95, while still maintaining a recall of over 0.95. And the F2 score, a weighted harmonic mean of the precision and recall that gives more weight to recall than precision, of both algorithms were able to achieve over 0.97. This is significant because it means that the algorithms were able to identify a high percentage of fraudulent transactions, while also avoiding the false positives that can be common with other fraud detection methods.

We found that SMOTEENN hybrid sampling method worked best with RF and XGBoost. SMOTE-TOMEK and SMOTE sampling method had a similar result compared to the original dataset without any resampling, whereas ADASYN, random over sampling, and random under sampling had a worse result compared to the original dataset. This is also expected because hybrid sampling methods over-sample first to create synthetic data but also undersample afterwards to reduce noise to provide better training.

The results of the study showed that both random forest (RF) and XGBoost are the least prone to overfitting. This is because the testing precision-recall curves for both algorithms were similar to the training precision-recall curve. This makes sense because RF and XGBoost are both ensemble methods that combine the predictions of multiple models to create a more accurate and robust model. In addition, RF and XGBoost both use bagging, which means that they train each model on a different subset of the training data. This helps to prevent the models from becoming too closely correlated with each other, which can also lead to overfitting.

The study also found that the SMOTEENN hybrid sampling method worked best with RF and XGBoost. SMOTEENN is a hybrid sampling method that combines SMOTE (Synthetic Minority Oversampling Technique) and ENN (Edited Nearest Neighbors). SMOTE creates synthetic data points for the minority class, while ENN removes data points that are close to the decision boundary. This helps to improve the accuracy of the model by reducing the bias in the training data. The study also found that SMOTETOMEK and SMOTE sampling methods had similar results to the original dataset without any resampling. ADASYN, random over sampling, and random under sampling had worse results than the original dataset. This is because these sampling methods can introduce noise into the training data, which can lead to overfitting.

Our current work has some limitations. The large dataset requires a lot of computational time and resources to train. Even though GridSearchCV can be very helpful in model selection, the computation time required to obtain the ideal values is extensive, especially for a large dataset like this. This is why we use RandomizedSearchCV instead to be effective in finding good hyperparameter values.

To ensure the prediction of credit fraud is fast and accurate, we can adapt the transfer learning approach. This means that we will use our pre-trained ML model from this study and train it on a randomly selected small number of equal normal and fraudulent transactions from new datasets. Because the sample size of the new dataset is much smaller, the required training time and computational memory requirement becomes less. This will allow us to achieve real-time prediction for credit transactions in the future while still ensuring the correctness of the prediction.

In addition to the current preprocessing methods, we can also adopt other methods such as outlier removal to reduce the sample size while making the model easier to predict. We can also apply other feature engineering approaches. Currently, the dataset is transformed using Principal Component Analysis (PCA) to prevent sensitive information leakage. Feature importance ranking using Random Forest (RF) or Extreme Gradient Boosting (XGBoost) can be used to reduce feature size and reduce training time and memory.

In the future, we can also apply newer neural network-based machine learning models. This includes building a hybrid convolutional neural network and long short-term memory (CNN-LSTM) framework. CNNs are capable of learning feature representations without the need for feature engineering, while LSTMs are known for their feedback connections that can learn long-term dependencies (Gajamannage & Park, 2022; Le Borgne et al.). We will then be able to compare model performance of the combined deep learning algorithm with RF and XGB.

References

Sukarna Barua, Md Monirul Islam, and Kazuyuki Murase. A novel synthetic minority oversampling technique for imbalanced data set learning. In *Neural Information Processing: 18th International Conference, ICONIP 2011, Shanghai, China, November 13-17, 2011, Proceedings, Part II 18*, pp. 735–744. Springer, 2011.

Gustavo EAPA Batista, Ronaldo C Prati, and Maria Carolina Monard. A study of the behavior of several methods for balancing machine learning training data. *ACM SIGKDD explorations newsletter*, 6(1):20–29, 2004.

Jakob Brandt and Emil Lanzén. A comparative review of smote and adasyn in imbalanced data classification. 2021.

Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

Kelum Gajamannage and Yonggi Park. Real-time forecasting of time series in financial markets using sequentially trained many-to-one lstms, 2022.

Haibo He, Yang Bai, Edwardo A Garcia, and Shutao Li. Adasyn: Adaptive synthetic sampling approach for imbalanced learning. In *2008 IEEE international joint conference on neural networks (IEEE world congress on computational intelligence)*, pp. 1322–1328. IEEE, 2008.

Yann-Aël Le Borgne, Wissam Siblini, Bertrand Lebichot, and Gianluca Bontempi. *Reproducible Machine Learning for Credit Card Fraud Detection - Practical Handbook.*

Roweida Mohammed, Jumanah Rawashdeh, and Malak Abdullah. Machine learning with oversampling and undersampling techniques: overview study and experimental results. In *2020 11th international conference on information and communication systems (ICICS)*, pp. 243–248. IEEE, 2020.

Ivan Tomek. Two modifications of cnn. 1976.

Dennis L Wilson. Asymptotic properties of nearest neighbor rules using edited data. *IEEE Transactions on Systems, Man, and Cybernetics*, (3):408–421, 1972.

Zhuoyuan Zheng, Yunpeng Cai, and Ye Li. Oversampling method for imbalanced classification. *Computing and Informatics*, 34(5):1017–1037, 2015.