Dealing with Imbalanced Datasets

En-Chun, KUO
111423052
daisykuo030@g.ncu.edu.tw

Zi-Rong, WANG
112423070
112423070@g.ncu.edu.tw

Hsuan-Chu, WANG
112423072
112423072@g.ncu.edu.tw

ABSTRACT

The main objective of this study is to review the literature and explore issues related to imbalanced data, including: (1) What constitutes an imbalanced dataset? (2) What impacts do imbalanced datasets have? (3) Methods for handling imbalanced datasets, and (4) Evaluation metrics. The study will conduct experiments based on the methods mentioned in the literature for handling imbalanced datasets, discussing the effectiveness of different methods in improving imbalanced data.

Keywords

Data imbalance, Classification, Data-level methods, Algorithms methods, Hybrid methods, Credit Card Fraud detection, Ensemble

1. INTRODUCTION

1.1 What is a class imbalanced dataset?

Handling the imbalanced data is not an easy task in computational process. Imbalance refers to the unequal number of data points in different classes. If the number of samples in one class in a dataset is higher than the others, this class is said to be a "majority class". In other words, if the number of samples in one class is lower than the others in the same dataset, it is said to be a "minority class"; such type of dataset is known as an "imbalanced dataset"[1].

Examples of typical imbalanced datasets include email classification problems, credit fraud classification problems, etc. Taking email classification as an example, since the number of emails classified as spam is much smaller than those classified as legitimate emails, the original distribution of samples in the two classes leads to dataset imbalance. In this scenario, the predictive accuracy for the majority class will be much higher than that for the minority class, as the classifier needs to ignore samples from the majority of legitimate emails to achieve maximum accuracy[2].

1.2 Why is class imbalance worth investigating?

Class imbalance issues are prevalent in many real-world datasets, leading models to favor majority classes and perform poorly on minority classes. Such misclassifications can yield dubious outcomes, particularly in critical applications like disease diagnosis, prompting researchers to prioritize addressing class imbalance.

Learning from skewed datasets becomes crucial in numerous practical classification scenarios, such as fault prediction, fraud detection, medical diagnosis, text classification, oil spill detection in satellite images, and cultural modeling. In these cases, the costs associated with misclassification for each class are unequal. For instance, in software fault prediction, missing a defect incurs a significantly higher cost than a false-positive error during the testing phase of software development.[3]

Moreover, data distributions among classes in many real-world situations are non-uniform, with at least one class having fewer samples than others. This poses a challenge for standard machine learning algorithms, as they may exhibit bias towards majority classes, resulting in inferior performance on minority classes.

Addressing these imbalance issues is crucial for enhancing model performance and accuracy. Researchers have proposed various methods, including data-level and algorithm-level approaches, to tackle these challenges. Understanding and resolving class imbalance problems can improve the reliability and effectiveness of machine learning models in real-world applications.

1.3 The Scope of our Survey

Imbalanced data is a critical issue. There are three categories of methods to address imbalanced datasets: data-level methods, algorithm-level methods, and hybrid methods. [1] This paper will investigate the aforementioned three methods and experimentally discuss their effectiveness in handling imbalanced datasets.

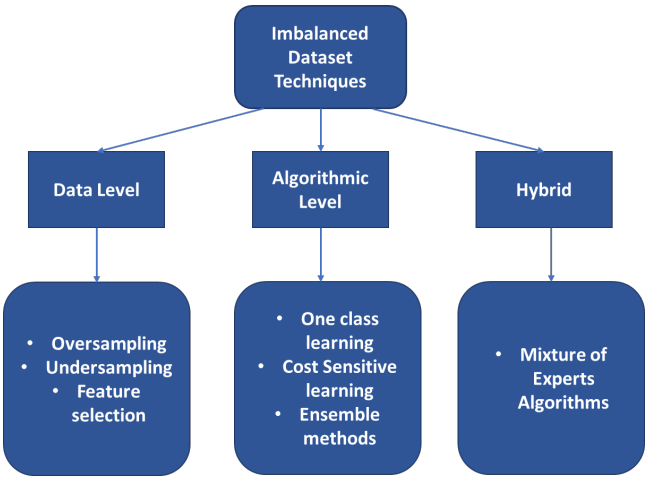

Figure 1. Methods to handle the imbalanced data

2. HANDLING IMBALANCED DATA PROBLEM

In this section, we have organized methods for handling imbalanced datasets based on our search criteria, which include three types of approaches: Data-level methods, Algorithm-level methods, and Hybrid methods. Subsequently, we introduce evaluation metrics for assessing imbalanced datasets, followed by an overview of the characteristics inherent to imbalanced datasets.

2.1 Methods

2.1.1 Literature Search Procedure

To systematically understand methods for dealing with imbalanced datasets, we first conducted a literature search using the keyword "Imbalanced dataset" to comprehensively gather relevant information about imbalanced datasets. Additionally, to conduct the empirical experiments, we selected a dataset from the financial domain. Therefore, we also conducted a search for "financial imbalanced datasets." This process involved utilizing various academic journals, papers, and other trustworthy sources of information.

Simultaneously, we organized the solutions mentioned in the literature and used these methods as keywords to search for related literature. This ensured that we had sufficient understanding of how these methods operate to smoothly proceed with experiments.

2.1.2 Data-level methods

Data-level methods tackle the imbalance problem at the data level by removing or replicating samples to balance the dataset. They are often referred to as external methods [1]. These methods can be categorized into Oversampling, Undersampling, Oversampling followed by Undersampling, and feature selection [2].

2.1.2.1 Oversampling Methods

Also known as upsampling, it is a sampling technique that balances the dataset by replicating samples from the minority class without losing the original data. However, it may lead to issues such as overfitting and increased computational complexity due to the enlarged dataset.

Oversampling methods are categorized into two types: random oversampling method and informative sampling method. The random oversampling method involves randomly selecting samples from the minority class for replication, while the informative sampling method synthesizes minority class data points based on predefined criteria.

Random oversampling (ROS), Synthetic minority oversampling technique (Smote), and Adaptive synthetic sampling (AdaSyn) are all oversampling methods.

(a) Random oversampling (ROS)

ROS is an effective and widely used oversampling approach for handling class imbalance. It randomly selects minority class samples and replicates them. Although efficient, random selection may increase the likelihood of selecting multiple instances from the same minority class, leading to potential overfitting [2],[6]

(b) Synthetic minority oversampling technique (Smote)

Smote is a popular oversampling technique for class imbalance. Based on the KNN algorithm, it randomly selects a small sample and its K nearest neighbors of the same class, then generates a new sample by selecting one of the K neighbors and interpolating between it and the selected sample. While Smote works effectively for smaller datasets, it may become inefficient for larger datasets due to the time required for synthesizing data points, increased overlap between data points, and inability to accurately reflect the original data distribution.[4]

(c) Synthetic sampling (AdaSyn)

AdaSyn is an improved algorithm based on SMOTE. It addresses the limitation of SMOTE in generating overlapping data points. AdaSyn automatically determines the number of synthetic samples needed for each minority class sample based on the distribution of the dataset, assigning higher weights to complex or difficult-to-learn samples. Unlike Smote, which generates the same number of synthetic samples for each minority sample, AdaSyn's adaptive approach improves dataset balance, reduces learning bias, but may be susceptible to noise interference.[5]

2.1.2.2 Undersampling methods

These are sampling techniques that achieve dataset balance by removing samples from the majority class, leading to potential loss of valuable data. They are suitable when the dataset is sufficiently large.[6]

(a) Random undersampling (RUS)

RUS is the most common and efficient undersampling technique. It deletes examples from the majority class and can result in losing information invaluable to a model.However, its main drawback is the potential loss of important data.

(b) Tomek links

If the two nearest samples belong to different classes, they form a Tomek link. One of the samples in the Tomek link may be noise or outliers, or both samples may lie on the boundary. By removing Tomek links, samples overlapping between different classes are deleted, ensuring that nearest neighbors belong to the same class, thereby improving classification.[7]

(c) Edited Nearest Neighbours (ENN)

If a sample from the majority class has more than half of its K nearest neighbors belonging to a different class, it is removed.

2.1.2.3 Oversampling followed by Undersampling methods

This approach addresses the different advantages, disadvantages, and limitations of oversampling and undersampling techniques.

(a) Smote + Tomek (SmoteTomek)

Firstly, Smote is applied, which may lead to overfitting. Then, Tomek is used to eliminate this overfitting, achieving a balanced dataset and improving classifier performance.

(b) Smote + ENN (Smoteen)

Initially, ENN is employed to remove samples different from their neighbors. Subsequently, Smote is used to synthesize data, balancing the dataset.

2.1.2.4 Feature selection methods

Feature selection methods involve selecting features from original data without altering the original features. To address this challenge, Liuzhi Yin et al [5] proposed two novel feature selection techniques. The first approach is based on class decomposition, where large classes are initially divided into smaller pseudo-subclasses. Subsequently, feature selection is performed on the new subset of data to assess the quality of features. The second method relies on Hellinger distance for feature selection. This approach disregards class prior information when computing distances, thereby demonstrating greater tolerance towards skewed class distributions [8], [9].

2.1.3 Algorithm-level methods.

Unlike data sampling methods, algorithmic methods for handling class imbalance do not alter the distribution of training data. Instead, the learning or decision-making process is adjusted in such a way as to increase the importance of the positive class. The most common practice involves modifying the algorithm to take intoaccount class penalties or weights, or shifting the decision threshold to reduce bias towards the negative class. Algorithmic level methods are often referred to as an internal approach because they involve the design of new classification algorithms or the enhancement of existing algorithms to address the bias introduced by imbalanced data. Algorithmic level methods can mainly be divided into cost-sensitive, ensemble-based, and threshold methods.[3], [10]

2.1.3.1 Cost-sensitive Learning

Assign different costs to misclassifications of different classes, making the algorithm pay more attention to the classes with higher costs during the training process (usually the minority classes). This approach is achieved by adjusting the loss function, with the main goal of reducing the overall cost.[11]

The common applications of cost-sensitive learning are in various classification problems, especially in contexts such as financial fraud detection and medical diagnosis, where the consequences of errors are notably significant.

(a) AdaCost

AdaCost retains AdaBoost's structure but incorporates a cost factor for updating weights, emphasizing errors with higher costs, typically those from the minority class. This adjustment not only considers the correctness of classification but also the error's cost, potentially increasing the weight of minority class samples regardless of their classification accuracy to prioritize high-cost errors in further iterations.[12]

Advantages: AdaCost directly targets and aims to reduce high-cost misclassifications, enhancing the recall rate for the minority class in certain scenarios.

Limitations: Fine-tuning cost parameters to the specific problem requires detailed adjustments, likely involving thorough testing and deep understanding to find the best settings.

AdaC1, AdaC2, and AdaC3 are algorithms that build on AdaCost to address imbalanced data by adjusting weights for misclassified samples:

(b) AdaC1

AdaC1 boosts attention to misclassified minority samples by increasing their weights based on classification accuracy.

(c) AdaC2

AdaC2 refines this approach by considering classification confidence, especially for samples close to the decision boundary, potentially enhancing performance.

(d) AdaC3

AdaC3 merges AdaC1 and AdaC2's strategies, adjusting weights by both accuracy and confidence for a more comprehensive solution.

These methods offer several advantages in enhancing the recognition of minority classes by adjusting sample weights, thereby aiming to achieve a better balance between minority class detection and overall model performance. [11] However, their effectiveness heavily relies on accurately tuning sample weights, which can increase computational demands. Optimizing these adjustments for practical use may require extensive experimentation.

2.1.3.2 Ensemble Methods

Ensemble methods, also known as multiple classifier systems, enhance the performance of learning methods by combining a set of base classifiers within the classification system. They are among the most frequently employed classifiers for addressing the issue of imbalanced data. The output of each base classifier is aggregated and used to make the classification decision for new samples.[13] The primary objective of ensemble methods is to achieve better predictive performance compared to using a single classifier. Bagging and boosting are two well-known techniques within ensemble classifiers.[14], [15]

In bagging, the original training set is partitioned into N subsets of equal size, and each subset is utilized to construct a classifier. The complete classification model is then built by aggregating these individual classifiers. In contrast, boosting algorithms produce a series of weak learning models, and through multiple iterations, these algorithms combine these weak learners into a single prediction model that is significantly more accurate than any individual weak learner.

Several studies have demonstrated that ensemble methods are effective, especially when dealing with datasets containing high-dimensional features, as they enhance prediction accuracy and stability by aggregating multiple models.[16] Specifically for imbalanced data, ensemble methods such as Random Forest and Boosting Trees can enhance the focus on minority data by leveraging the combination of multiple weak learners.

(a) AdaBoost

AdaBoost is one of the many boosting algorithms used to address the problem of weak learners. AdaBoost (Adaptive Boosting) is a machine learning strategy of ensemble methods that creates a strong classifier by combining multiple weak classifiers. This variant of boosting can tackle both classification and regression problems. AdaBoost achieves efficient data classification and regression analysis by iteratively enhancing simple decision rules.[17][18]

(b) MEBoost

It is an improved Boosting method that enhances the model's classification performance on and near the margin samples. In addressing imbalanced datasets, MEBoost enhances performance by combining two different weak learners with boosting techniques, offering an alternative to existing technologies such as SMOTEBoost, RUSBoost, and Adaboost. The efficacy of MEBoost has been assessed across 12 benchmark imbalanced datasets, compared with state-of-the-art ensemble methods like SMOTEBoost, RUSBoost, Easy Ensemble, EUSBoost, and DataBoost. Experimental results demonstrate significant improvements over other methods, concluding that MEBoost is an effective and promising algorithm for handling imbalanced datasets.[19]

2.1.3.3 Threshold-moving

Adjust the decision threshold for classification to give more preference to minority classes when predicting. This is usually

done after the model training is complete, adjusting based on the desired performance metrics. This technique is well-suited for situations necessitating heightened sensitivity in identifying minority classes, such as disease and fraud detection.[20]

Compared to cost-sensitive methods, threshold-moving is a post-processing strategy that does not alter the model training process. It offers greater flexibility, as it can be directly applied to any pre-trained model without the need for retraining. On the other hand, cost-sensitive learning integrates different costs with varying weights directly into the model's training process, guiding the learning path. Modifications in this approach may necessitate changes to the original model.

2.1.4 Hybrid methods

While both data-level methods and algorithm-level methods techniques aim to address class distribution imbalances, they each bring along their own set of drawbacks such as over-generalization and the potential loss of valuable data.[1] In response to these challenges, ensemble learning has emerged as one of the most commonly used classifiers, which integrates data-level and algorithmic-level methods to effectively address the issue of imbalanced data. [5] By combining these two types of methods, data-level methods can adjust the data distribution to reduce the degree of data imbalance, while algorithm-level methods can enhance the learning process to improve classification performance.

In various fields, numerous papers have proposed hybrid methods to address imbalanced datasets. We have selected several hybrid methods, which are introduced as follows.

In 2008, Seiffert et al. introduced RUSBoost, an algorithm designed to address class imbalance issues in datasets with discrete class labels. It employs a combination of random under-sampling (RUS) and the AdaBoost boosting procedure to improve modeling of the minority class by removing samples from the majority class. RUSBoost demonstrates a higher average performance across all seven datasets, achieving an AUC of 0.8991 compared to SMOTEBoost's 0.8816. [21], [22]

R. Alejo et al. proposed combining the move method MBP (modified back-propagation) with GGE (Gabriel Graph Editing) algorithm, and through experiments, they demonstrated that this approach is more effective than SMOTE + GGE in simultaneously addressing the issues of class imbalance and class overlap.

EUSBoost, proposed by Mikel Galar et al. in 2013, builds upon RUSBoost by incorporating Evolutionary Undersampling (EUS) alongside bagging and boosting techniques, leading to enhanced classification accuracy.[23]EUSBoost, proposed by Mikel Galar et al. in 2013, builds upon RUSBoost by incorporating Evolutionary Undersampling (EUS) alongside bagging and boosting techniques, leading to enhanced classification accuracy.[23]

In the recent year, Krawczyk introduced a novel strategy for addressing imbalanced multi-class datasets by merging pairwise One-Versus-One (OVO) decomposition with ensemble learning. This involved splitting the original dataset into multiple binary sub-problems to streamline the classification process. Subsequently, an ensemble classifier tailored for handling these simplified binary tasks was employed, and the outputs of individual classifiers were amalgamated using a pairwise coupling technique.

Gong et al.'s RHSBoost employs a random hybrid sampling approach that blends random undersampling with ROSE sampling techniques. This maintains the original dataset size while balancing class distribution, potentially avoiding the pitfalls of model overfitting and excessive training time. However, the sequential use of two sampling methods adds complexity and processing time.[24]

Also, Zhao et al. (2020) showcased a weighted hybrid ensemble method that employs a combination of two sampling methods alongside two base classifiers.[25]

In summary, most of these hybrid methods are based on ensemble methods, adjusting the training set through data-level techniques before model refinement and learning. They showcase a promising path towards enhancing classification performance in the context of imbalanced data, providing a variety of strategies to tackle class imbalances and strengthen model robustness.

Furthermore, drawing upon various methodological approaches discussed above, it can be broadly concluded that the most prevalent techniques currently employed encompass integrated methods combined with preprocessing of training set data, followed by further enhancement through boosting methodologies for classifier learning. Researchers have indicated that numerous hybrids boosting methods exhibit significant efficacy not only for addressing imbalanced datasets but also demonstrate notable performance across diverse dataset types, as evidenced by algorithms such as CatBoost and XGBoost.**[12], [26], [27]**

Table 1. Overview of the approaches dealing with imbalanced datasets

Author	Title of Paper	Brief Description	Method	Year
Chawla et al.	SMOTE: Synthetic Minority Over-sampling Technique	The paper proposes a method for constructing classifiers from imbalanced datasets by combining over-sampling of the minority class with under-sampling of the majority class.	Data-Level	2002
Freund, Yoav	A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting	This paper presents a new boosting algorithm (AdaBoost) for dynamic resource allocation in online scenarios and extends the approach to handle non-binary outcomes without requiring prior	Algorithm-level	1997

		knowledge of weak learners' performance.		
F. Rayhan *et al.*	MEBoost: Mixing Estimators with Boosting for Imbalanced Data Classification	Introduces a boosting algorithm designed for imbalanced datasets, combining two weak learners to outperform. Results show MEBoost's effectiveness in handling class imbalances.	Algorithm-level	2017
J. Tanha, Y. Abdi, N. Samadi, N. Razzaghi, and M. Asadpour	Boosting methods for multi-class imbalanced data classification: an experimental review	This paper evaluates boosting techniques for multi-class imbalanced datasets, finding CatBoost and LogitBoost most effective and recommending MMCC as the optimal evaluation metric.	Algorithm-level	2020
Seiffert et al.	RUSBoost: Improving classification performance when training data is skewed	RUSBoost, a combination of random under-sampling (RUS) and the AdaBoost boosting procedure.	Hybrid	2017
He et al.	ADASYN: Adaptive synthetic sampling approach for imbalanced learning	The method combines ensemble learning with sampling techniques and cost-sensitive learning to achieve better classification outcomes.	Data-Level	2008
R. Alejo et al.	A hybrid method to face class overlap and class imbalance on neural networks and multi-class scenarios	The method combines modified back-propagation (MBP) with the Gabriel Graph Editing (GGE) algorithm	Hybrid	2013
Krawczyk	Combining One-vs-One Decomposition and Ensemble Learning for Multi-class Imbalanced	Proposed the method combining One-Versus-One (OVO) decomposition with ensemble learning.	Hybrid	2016
Gong and Joonho	RHSBoost: Improving classification performance in imbalance data	RHSBoost, combining random undersampling with ROSE sampling techniques.	Hybrid	2017
Zhao et al.	A weighted hybrid ensemble method for classifying imbalanced data	Proposed a novel method combining of two sampling methods with two classifiers	Hybrid	2020

2.2 Imbalanced Datasets in the Financial Field

Imbalanced datasets are prevalent in the financial sector, particularly in areas such as financial fraud detection, precision marketing analysis in finance, and credit default prediction. These domains frequently encounter the challenge of imbalanced categories within the data.

Serveral studies explore experiments using various types of imbalanced financial datasets. Veganzones and Severin explored bankruptcy prediction in imbalanced datasets, employing the Support Vector Machine method and employing SMOTE to address dataset imbalance.[28] Sahin et al. introduced a decision tree classification algorithm incorporating cost-sensitive learning techniques, specifically for financial fraud detection.[29] Yu also showcased the combined algorithm for classifying imbalanced datasets in the financial field, providing an effective approach for dataset classification.[30]

In this research, we will conduct the empirical experiment with the financial dataset in the later section.

2.3 Evaluation metrics

In the realm of machine learning, evaluation metrics play a crucial role in assessing the performance and efficacy of models, particularly when dealing with imbalanced datasets. Imbalanced data poses a unique challenge in classification tasks, where certain classes are underrepresented compared to others. In this section, we gather the primary evaluation metrics sourced from various studies utilized for addressing imbalanced data, elucidating their significance in model evaluation. We will introduce three categories of performance metrics: Single-class metrics, Overall metrics and statistical comparison.[12]

2.3.1 Single-class metric

2.3.1.1 Precision

Precision measures the proportion of true positive samples correctly predicted by the model out of all samples predicted as positive. It is calculated using the formula as below.

$$\textbf{\textit{Precision} = TP / (TP + FP)}$$

In the context of imbalanced data, precision helps in evaluating the model's ability to make accurate positive predictions, especially when the positive class is a minority class.

2.3.1.2 Recall

Recall, also known as sensitivity, quantifies the model's capability to identify all relevant instances in a dataset. It is calculated as:

$$Recall = TP / (TP + FN)$$

FN represents false negatives. Recall is crucial in imbalanced datasets to ensure that the model can capture all positive instances effectively, minimizing false negatives.

2.3.1.3 F1-Score

In imbalanced data settings, there exists a trade-off between precision and recall. To strike a balance between these metrics, the F-measure is utilized, which is the harmonic mean of precision and recall:

$$F\text{-}Measure = (2 * Precision * Recall) / (Precision + Recall)$$

By adjusting the beta parameter, users can emphasize either precision or recall based on the specific requirements of the classification task on imbalanced data.

2.3.1.4 Geometric Mean (G-Mean)

The geometric mean, or G-mean, serves as a suitable metric for evaluating models on imbalanced datasets. It considers the balance between sensitivity and specificity, making it particularly valuable in scenarios with imbalanced class distributions.

$$G\text{-}Mean = sqrt(Sensitivity * Specificity)$$

2.3.1.5 Accuracy

Accuracy is a fundamental metric that calculates the proportion of correctly classified instances out of the total number of instances in a dataset. It is defined as:

$$Accuracy = (TP + TN) / (TP + TN + FP + FN)$$

2.3.1.6 Balanced Accuracy

Balanced Accuracy is calculated as the average of sensitivity (TP) and specificity (TN) of a classifier. A model that predicts the majority class with high accuracy may still perform poorly in identifying the minority class, leading to skewed results. Balanced Accuracy addresses this issue by considering the performance across all classes equally, giving a more accurate representation of the model's overall effectiveness.

$$Balanced\ Accuracy = (TP + TN) / 2$$

2.3.2 Overall metrics

2.3.2.1 ROC Curve

The Receiver Operating Characteristic (ROC) curve is a graphical representation of a classifier's performance across various threshold settings.[31] It illustrates the trade-offs between true positive and false positive rates, offering a comprehensive view of the model's performance on imbalanced datasets. [31] It illustrates the trade-offs between true positive and false positive rates, offering a comprehensive view of the model's performance on imbalanced datasets. The ROC curve shows all possible conflicts between true-positive rate (TPR) and false-positive rate (FPR) across various decision thresholds. A well-defined ROC curve indicates the model's ability to distinguish between classes accurately

2.3.2.2 Area Under the ROC Curve (AUC)

The Area Under the ROC Curve (AUC) quantifies the overall performance of a classifier by calculating the area under the ROC curve. In the context of imbalanced data, AUC is a critical metric for assessing the model's ability to discriminate between classes effectively. AUC evaluation metric converts this curve to a value in the range of [0.5, 1], where the value 1 shows a perfect classifier and the lower value 0.5 means the classifier does not work better than random guess. A higher AUC value signifies superior model performance in handling imbalanced datasets.[32][32]

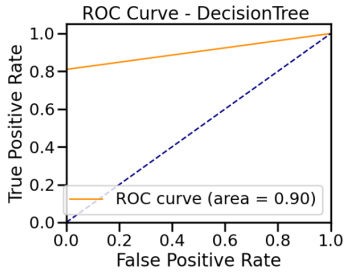

Figure 2. The Example Plot of ROC Curve

2.3.2.3 Matthews Correlation Coefficient (MCC)

The Matthews Correlation Coefficient (MCC), introduced by biochemist Brian W. Matthews in 1975, is particularly utilized in the analysis of imbalanced binary class datasets. [33] It incorporates the values of all classes in the confusion matrix to compute a correlation measure between actual and predicted samples. The MCC's range extends from -1 to +1, where a value of +1 indicates perfect prediction, while -1 denotes inverse prediction. It is calculated as:

$$MCC = (TP * TN - FP * FN) / sqrt((TP + FP) * (TP + FN) * (TN + FP) * (TN + FN))$$

2.3.3 Statistical Comparison

In certain studies, a statistical test suite is utilized to validate results and discern significant differences among them. This comparison is commonly classified into parametric and nonparametric test methods. [12], [34]

2.3.3.1 Parametric Tests

Parametric tests, such as the paired t-test and ANOVA, are commonly used for comparing classifier results. However, these tests rely on strict assumptions about the data distribution, which may not always hold true in machine learning studies on imbalanced data, leading to unreliable results.

2.3.3.2 Nonparametric Tests

Nonparametric tests, including the Wilcoxon and Friedman tests, offer a more flexible approach to comparing classifier performance on imbalanced datasets. These tests do not assume a specific data distribution, making them suitable for diverse machine learning scenarios. Nonparametric tests are preferred when the assumptions of parametric tests are not met, ensuring robust and reliable comparisons among classifiers.

3. Discussion

In this section, we introduce the selected experimental methods and highlight potential differences between them.

3.1 Data-Level Methods

3.1.1 Model Selection

In model selection, we chose one model for the ensemble classifier and one model for the base classifier to conduct experiments. In the literature under consideration, the Random Forest model is mentioned as an ensemble classifier composed of multiple decision trees, which achieves higher accuracy in imbalanced datasets [4]. Therefore, we selected **Random Forest** as an experimental model for the data-level method. For the base classifier selection, we followed the suggestion from the scikit-learn algorithm cheat-sheet and opted for a suitable classifier for large datasets and non-text data, namely **KNN(K Nearest Neighbor)**, to conduct experiments.

3.1.2 Sampling Method Selection

We chose the Oversampling method, **Smote**, and the Undersampling method, **TomekLinks**, as they exhibited superior average performance in the literature review. Additionally, we included the two previously discussed methods, **Smote+TomekLinks** and **Smote+ENN**, which combine Oversampling followed by Undersampling techniques, for sample processing.

3.2 Algorithm Methods

3.2.1 Model Selection

In the selection of methods, we employed ensemble learning approaches, which construct a strong learner by combining multiple weak learners. This usually achieves higher predictive accuracy than a single model because it can learn the characteristics of data from various perspectives, enhancing the capability to recognize a minority of samples [35]. Among them, we chose AdaBoost and MEBoost.

3.2.2 AdaBoost

AdaBoost is a widely used ensemble learning algorithm that iteratively adjusts sample weights to increase the model's focus on difficult-to-classify samples. Although not specifically designed for handling imbalanced data, its weight adjustment mechanism allows it to pay more attention to minority class samples to some extent, providing a basis for further improvements aimed at imbalanced datasets.

3.2.3 MEBoost

MEBoost is an improved version of AdaBoost, specifically optimized for imbalanced datasets. It is designed to deal with imbalanced data sets, adding model diversity and a resampling mechanism compared to AdaBoost, making it an ideal choice for addressing imbalanced data issues. Therefore, we observed the impact of algorithmic improvements on model discriminative performance by comparing the different scores of AdaBoost and MEBoost on the same dataset.

3.3 Hybrid Methods

3.3.1 Model Selection

Within the category of hybrid methods, we have chosen to focus on RUSBoost and RHSBoost. RUSBoost emerged as one of the early approaches for handling imbalanced datasets.[22] Later on, RHSBoost was introduced as an improvement upon RUSBoost by refining the sampling technique.[24] Thus, the motivation behind selecting these methods lies in exploring the differences between the two approaches, allowing for a comparative analysis of their effectiveness and advancements over time. The following will provide a detailed introduction to the algorithms.

3.3.2 RUSBoost

3.3.2.1 Initialization

The algorithm begins by initializing the weights of each example in the training dataset. If there are m examples, each example is assigned an initial weight of $1/m$.

3.3.2.2 Iterative Training

RUSBoost iteratively trains T weak hypotheses. Each iteration consists of the following steps.

(a) Random Undersampling

A temporary training dataset is created by randomly removing examples from the majority class until the minority class represents a specified percentage (N%) of the new dataset. This results in a new weight distribution for the examples.

(b) Weak Learner Training

The weak learner is provided with the examples from the undersampled dataset and their corresponding weights.

(c) Hypothesis Generation

The weak learner returns a hypothesis h_t, which is a function that maps input features X and class labels Y to a probability in the range [0, 1].

(d) Weight Update Parameter Calculation and Weight Update

The weight update parameter 'αt' is calculated using the formula $\alpha_t = 1/2 * log((1 - \varepsilon_t) / \varepsilon_t)$. This parameter will be used to adjust the weights of the examples for the next iteration. The weights for the next iteration D_t+1 are updated using the current weights D_t, the hypothesis h_t, and the weight update parameter α_t.

3.3.2.3 Final Hypothesis Output

After T iterations, the final hypothesis $H(x)$ is formed as a weighted vote of all the weak hypotheses generated during the iterative process. The final classification decision for an example x is made by taking the arg_{max} over all class labels y in Y, summing the weighted votes from all hypotheses $h_t\ (x, y)$ and considering the weight update parameter α_t. The overall process is showed as Figure 3.

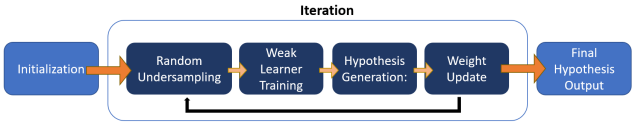

Figure 3. The RUSBoost Algorithm

3.3.3 RHSBoost

3.3.3.1 Initialization

The algorithm begins by initializing the weights of each example in the training dataset. If there are m examples, each example is assigned an initial weight of $1/m$. Additionally, smoothing matrices H_j for each class j are calculated. These matrices are used later in the weighted ROSE sampling step to generate new artificial data.

3.3.3.2 *Boosting Iterations*

(e) Undersampling

Perform random undersampling on the majority class of dataset using the current weight vector D_t to create a subset S_{under}.

(f) Weighted ROSE Sampling

This step is crucial in RHSBoost, where the weighted ROSE algorithm is applied to subset S_{under} using the current weight vector D_t. This step oversamples S_{under} until it reaches the size n, balancing the class distribution by adding newly sampled instances.

(g) Base Classifier Training

Train the base classifier h_t using the temporary balanced dataset $S_{temporary}$ obtained from the previous step.

(h) Weight Update

Calculate the error rate ϵt of the classifier h_t on the original dataset. The error rate is the sum of the weights $D_t(i)$ of the instances that h_t misclassifies. Then, calculate the coefficient βt which is used to update the weights and is defined as $\beta_t = 1 - \epsilon_t / \epsilon_t$. Then update the weights $D_t+1(i)$ for the next iteration, increasing the weights of the misclassified instances to focus on them in the next iteration.

3.3.3.3 *Final Classifier Construction*

After T iterations, the algorithm combines all the base classifiers h_t with their corresponding coefficients β_t to form the final classifier $H(x)$. This classifier makes predictions by taking a weighted vote among all the base classifiers. The overall process of RHSBoost Algorithm is shown as Figure 4, which is captured from the original paper.[24]

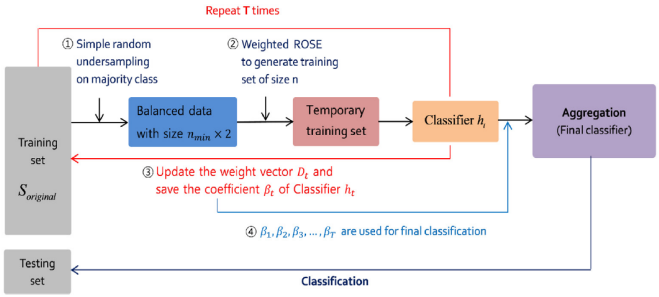

Figure 4. The RHSBoost Algorithm

The RHSBoost algorithm aims to create a strong classifier by focusing on the instances that are harder to classify, which are often the minority class instances in imbalanced datasets. The use of weighted ROSE sampling helps to generate synthetic instances that are representative of the minority class, thus addressing the imbalance issue.

3.3.4 *RUSBoost versus RHSBoost*

The significant difference between RHSBoost and RUSBoost lies in their approach to handling imbalanced data before training and their robustness to varying degrees of class imbalance. RUSBoost, integrating random undersampling with AdaBoost, shows a rapid decline in performance as the minority class ratio decreases. RHSBoost employs a hybrid sampling strategy that combines random undersampling with ROSE sampling, which is then integrated into the AdaBoost ensemble framework. This method is designed to maintain high classification performance even as the minority class ratio decreases. We will perform the empirical experiments in the next section.

4. EMPIRICAL EXPERIMENTS

In this section, we will begin by introducing the experimental design. Subsequently, we will conduct experiments on imbalanced datasets using methods categorized into data-level, algorithm-level, and hybrid-level approaches. Finally, we will discuss the experimental results

4.1 Experiments Design

In this experiment, we will conduct empirical experiments using various types of models selected from Section 4, and we will showcase the performance metrics of these models, the experiment design is shown **as Figure 6.**

We utilized the Credit Card Fraud Detection dataset obtained from kaggle.com (Credit Card Fraud Detection-Anonymized Credit Card Transactions Labelled as Fraudulent or Genuine, Machine Learning Group – ULB, 2018) [36]. We utilized the Credit Card Fraud Detection dataset obtained from kaggle.com (Credit Card Fraud Detection-Anonymized Credit Card Transactions Labelled as Fraudulent or Genuine, Machine Learning Group – ULB, 2018) [36].

The imbalance ratio is defined as the ratio between the numbers of instances in the majority class and minority class, which is first adopted in the work of Orriols-Puig and Bernadó-Mansilla (2009).[37] A higher value indicates a greater imbalance in the dataset. Regarding the imbalance ratio of the dataset used for the experiment, which is 577, it indicates that this dataset is highly imbalanced. The other characteristics of the datasets is shown as Table 2, and the category bar chart of the dataset is shown as Figure 5. In the following experiments, we split the dataset into a training set comprising 80% of the data and a testing set comprising 20% of the data.

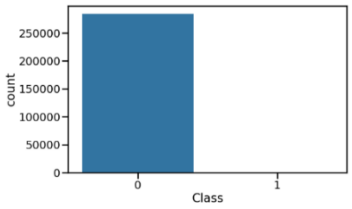

Figure 5. Original dataset samples: {0: 284315, 1: 492}

Table 2. Dataset characteristics

Name of the dataset	Credit Card Fraud Detection
Subject Area	Financial Data
Instances	284,807
Features	Anonymized features representing various transaction attributes (e.g., time, location, etc.)
Class	Binary label indicating whether the transaction is fraudulent (1) or not (0)
Missing Value	None
Data distribution	{0: 99.8%, 1: 0.2%}
Imbalance Ratio	577

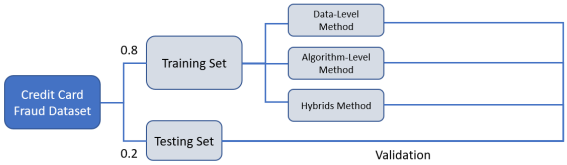

Figure 6. The Experiment Design

4.2 Data-level methods

4.2.1 Experimental Method

Models are built using the original samples as well as training set samples processed through Smote, Tomeklinks, Smote+Tomeklinks, and Smote+ENN for handling imbalanced data. The performance of classification is observed to determine any enhancements.

In the experiment, 80% of the dataset is used as training data. The sample numbers are as follows:

Table 3. Sample numbers of the dataset

Original complete samples dataset	**{0: 284315, 1: 492}**
Original training samples dataset	{0: 227451, 1: 394}
Smote Resampled dataset	{0: 227451, 1: 227451}
TomekLinks Resampled dataset	{0: 227387, 1: 394}
SmoteTomek Resampled dataset	{0: 227451, 1: 227032}
Smoteen Resampled dataset	{0: 221668, 1: 213835}

4.2.2 Results and analysis

From the experimental results (Table 6.), several important findings can be concluded:

4.2.2.1 Impact of Sampling Techniques

In terms of Recall, Balanced Accuracy, G-mean, and ROC-AUC, the performance of Random Forest model and KNN improved after employing sampling methods such as Smote, SmoteTomek, and Smoteen. This demonstrates that these methods, as indicated in the literature, are beneficial in addressing the issue of imbalanced datasets.

From the ROC-AUC curves (Figure 7., Figure 8.), it can be observed that at the same False Positive Rate, the ROC curves combined with sampling techniques have higher True Positive Rates compared to the ROC curves without using sampling techniques.

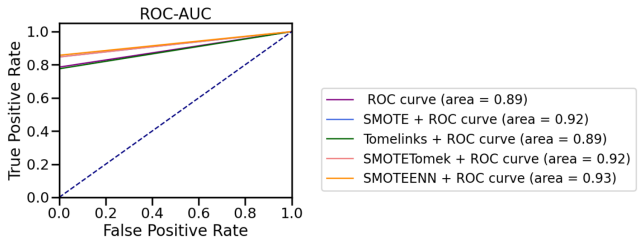

Figure 7. The ROC curve of the Random Forest model combined with sampling techniques.

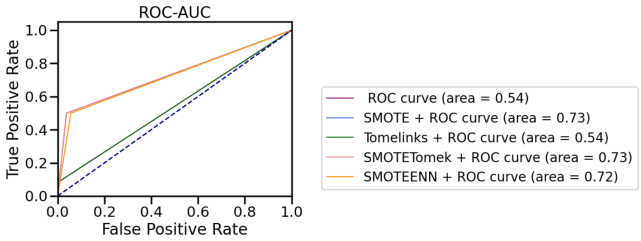

Figure 8. The ROC curve of the KNN model combined with sampling techniques.

4.2.2.2 Model Comparison

The performance of the Random Forest model surpasses that of the KNN model: Whether on the original dataset or the processed dataset, the Random Forest model demonstrates superior overall performance compared to the KNN model on this imbalanced dataset. This could be attributed to the significant impact of dataset imbalance on the KNN model.

4.2.2.3 Conclusion of Section 4.2:

In summary, the results from the experiments in Section 4.2 not only confirm that selecting appropriate data balancing techniques is beneficial for enhancing model performance when dealing with imbalanced datasets but also emphasize the importance of choosing suitable models. The overall better performance of the Random Forest model compared to KNN highlights the significance of selecting the right model.

4.3 Algorithm-level methods

4.3.1 Experimental Method

We use AdaBoost and MEBoost to test the selected imbalanced datasets. According to the methods described in the literature we referred to, incorporating the SMOTE technique prior to testing can yield better outcomes. Consequently, we also tested AdaBoost+SMOTE and MEBoost+SMOTE to identify the most suitable approach.

4.3.2 Results and Analysis

Based on the results (Table 6.), the following findings were obtained by comparing the performance of MEBoost and AdaBoost on imbalanced datasets:

4.3.2.1 Performance when use Individually

The results (Figure 9.) indicate that when used individually, MEBoost exhibits better performance compared to AdaBoost, specifically in terms of higher Balanced Accuracy and Recall. This highlights its advantage in achieving a high recall rate while maintaining reasonable precision. Although AdaBoost has a higher precision, its recall rate is relatively lower. This may be due to MEBoost being specifically designed to handle imbalanced datasets, allowing it to achieve a higher recall rate compared to the original algorithm.

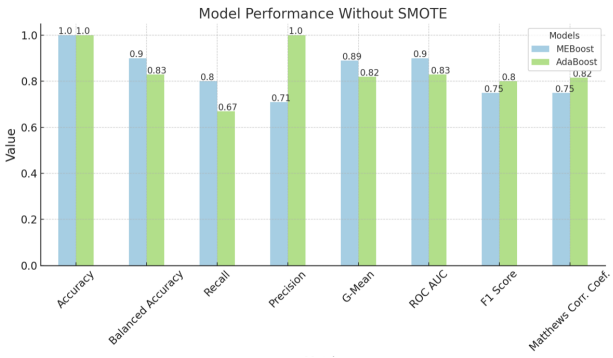

Figure 9. MEBoost vs AdaBoost Performance without SMOTE.

4.3.2.2 Performance Enhancement with SMOTE

After integrating SMOTE, both MEBoost and AdaBoost showed significant performance improvements. The enhancements in MEBoost's precision, F1 Score, and Matthews Correlation Coefficient (MCC) demonstrate an increase in its predictive power and reliability (Figure 10.). Simultaneously, there was a notable increase in AdaBoost's recall rate, suggesting that the incorporation of SMOTE has, to some extent, balanced the data distribution and enhanced the recognition of minority classes. Although there was a slight decrease in AdaBoost's accuracy, the increased recall rate had a positive impact on the model's performance.

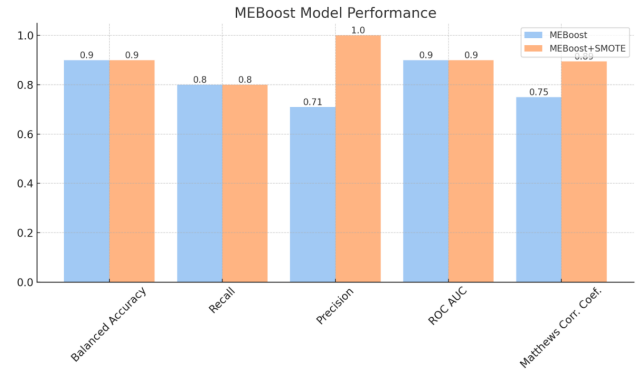

Figure 10. MEBoost Model Performance

4.3.2.3 Comprehensive Performance Assessment

In summary, MEBoost combined with SMOTE demonstrated superior performance on the datasets we tested, particularly in terms of balanced precision and recall. This proves that with proper resampling techniques, the performance of models on imbalanced datasets can be significantly improved. Without the use of SMOTE, even though AdaBoost performed better in precision, it was not as effective as MEBoost in recall, making it potentially less ideal for scenarios involving imbalanced datasets.

4.4 Hybrid methods

4.4.1 Experimental Method

We conducted experiments using the RUSBoost and RHSBoost hybrid methods to deal with imbalanced datasets, aiming to verify and compare whether RHSBoost performs better in handling imbalanced datasets. [18]

In the experiment, various base classifiers were selected, including the simple decision tree (which serves as the default classifier of the RUSBoost algorithm package), Random Forest Classifier, and Adaboost-trained classifier.

4.4.2 Results and Analysis

4.4.2.1 RUSBoost

In the RUSBoost Algorithm, the base classifier can be selected by the user. In our experiment, we utilize the simple Decision Tree Classifier, Random Forest Classifier and AdaBoost as the classifiers to observe the performance metrics. When the estimator's parameter specifies the number of base classifiers to be used, there can be a slight impact on the results when the estimator for the base classifier differs from the estimator for RUSBoost. The experiment result with applying several base classifiers is demonstrated as Table 4.

Throughout the experimental process, it was observed that the overall performance metrics were quite similar, for example, the balanced accuracy hovered around 0.94. Additionally, we attempted to set different values for the number of estimators. However, as the number increased, although there was a slight improvement in the MCC value, it led to a comparatively longer running time.

Furthermore, the number of estimators for the Decision Tree classifier did not exhibit significant differences in performance metrics. On the other hand, the Random Forest classifier performed the best in this experiment, demonstrating superior effectiveness in training the model as a base classifier.

Overall, the results suggest that while increasing the number of estimators can sometimes marginally improve MCC, it often comes at the cost of significantly increased computational time. There seems to be a point of diminishing returns where further increasing the number of estimators doesn't yield proportional improvements in performance but drastically increases computational resources. Therefore, it's essential to strike a balance between model performance and computational efficiency based on specific requirements and constraints.

Table 4. The experiment result of RUSBoost

Base Classifier	No. of Estimator Base Classifier	No. of Estimator RUSBoost	Balanced Acc.	MCC	Time
Decision Tree Classifier	50	10	0.94	0.21	3s
	50	50	0.94	0.21	12s
Random Forest Classifier	10	10	0.94	0.23	8s
	30	20	0.95	0.25	27s
	30	30	0.95	0.24	35s
	50	**10**	**0.94**	**0.25**	**20s**
	50	50	0.94	0.25	94s
AdaBoost Classifier	10	10	0.94	0.20	13s
	30	20	0.94	0.21	43s
	30	30	0.94	0.22	53s
	50	10	0.95	0.21	31s

	50	50	0.95	0.22	160 s

4.4.2.2 RHSBoost

In RHSboost, we conducted experiments using three types of base classifiers. It can be observed that when the decision tree's depth is set deeper, it exhibits better results, with balanced accuracy approaching 0.99. Similarly, the performance metrics presented by the random forest classifier are also promising, with the MCC value increasing from 0.18 to 0.26. However, there is a significant increase in computational time, especially when setting the base classifier as Adaboost classifier. The experiment result is shown in Table 6.

4.4.2.3 RUSBoost versus RHSBoost

Overall, in this dataset, RHSBoost outperforms RUSBoost. As shown in Table 5., the balanced accuracy in RHSBoost is nearly close to 0.99, and other performance metrics also surpass those of RUSBoost. However, due to RHSBoost's more extensive overall steps, it incurs significantly higher computational time. Therefore, considering the trade-off between computational resources and time, RUSBoost's performance in this dataset is also commendable.

Table 5. The experiment result of RHSBoost

Base Classifier	Iteration Times	Balanced Acc.	MCC	Time
Decision Tree (Max_Depth= 1)	30	0.94	0.18	37s
	40	0.94	0.18	67s
	50	0.95	0.17	138s
Decision Tree (Max_Depth= 5)	30	0.99	0.23	54s
	40	0.99	0.25	95s
	50	0.99	0.24	144s
Random Forest Classifier	30	0.99	0.26	94s
	40	0.99	0.26	135s
	50	0.98	0.26	154s
AdaBoost Classifier	30	0.98	0.18	2409s
	50	0.98	0.18	6678s

Table 6. The obtained results from Empirical Experiments

Type of Method	*method*	*Accuracy*	*Balanced Accuracy*	*Recall*	*Precision*	*G-mean*	*ROC AUC*	*F1-Score*	*MCC*
	Decision Tree Classifier	*1.00*	*0.89*	*0.89*	*0.86*	*0.89*	*0.89*	*0.88*	*0.76*
Data-level	Random Forest	***1.00***	*0.89*	*0.79*	*0.99*	*0.89*	*0.89*	***0.87***	***0.87***
	Smote+Random Forest	***1.00***	*0.92*	*0.85*	*0.94*	*0.92*	*0.92*	*0.86*	*0.86*
	Tomeklinks+ Random Forest	***1.00***	*0.89*	*0.78*	*0.99*	*0.88*	*0.89*	*0.86*	***0.87***
	SmoteTomek+ Random Forest	***1.00***	*0.92*	*0.85*	*0.94*	*0.92*	*0.92*	*0.86*	*0.86*
	Smoteen+ Random Forest	***1.00***	***0.93***	***0.86***	*0.93*	***0.93***	***0.93***	*0.86*	*0.86*
	KNN	***1.00***	*0.54*	*0.08*	***1.00***	*0.29*	*0.54*	*0.15*	*0.29*
	Smote+ KNN	*0.96*	*0.73*	*0.50*	*0.51*	*0.69*	*0.73*	*0.04*	*0.10*
	Tomeklinks+ KNN	***1.00***	*0.54*	*0.08*	***1.00***	*0.29*	*0.54*	*0.15*	*0.29*
	SmoteTomek+ KNN	*0.96*	*0.73*	*0.50*	*0.51*	*0.69*	*0.73*	*0.04*	*0.10*
	Smoteen+KNN	*0.94*	*0.72*	*0.50*	*0.51*	*0.69*	*0.72*	*0.03*	*0.08*
Algorithm-level	MEBoost	***1.00***	*0.90*	*0.80*	*0.71*	*0.89*	*0.90*	*0.75*	*0.75*
	MEBoost+SMOTE	*1.00*	*0.90*	*0.80*	*1.00*	*0.89*	*0.90*	*0.89*	*0.89*
	AdaBoost	*1.00*	*0.83*	*0.67*	*1.00*	*0.82*	*0.83*	*0.80*	*0.81*
	AdaBoost+SMOTE	*1.00*	*0.87*	*0.73*	*0.92*	*0.86*	*0.87*	*0.81*	*0.81*
Hybrid Method	RUSBoost w/ Decision Tree Classifier	*0.97*	*0.94*	*0.94*	*0.52*	*0.94*	*0.94*	*0.53*	*0.21*

	RUSBoost w/ Random Forest Classifier	*0.98*	*0.94*	*0.94*	*0.53*	*0.94*	*0.94*	*0.56*	*0.25*
	RUSBoost w/ Adaboost Classifier	*0.97*	*0.95*	*0.95*	*0.53*	*0.95*	*0.95*	*0.55*	*0.22*
	RHSBoost w/ Decision Tree Classifier	*0.97*	*0.99*	*0.99*	*0.53*	*0.99*	*0.98*	*0.55*	*0.24*
	RHSBoost w/ Random Forest Classifier	***0.98***	***0.99***	***0.99***	*0.53*	*0.99*	*0.98*	***0.56***	***0.26***
	RHSBoost w/ Adaboost Classifier	*0.95*	*0.98*	*0.98*	*0.52*	*0.98*	*0.97*	*0.52*	*0.18*

4.5 Overall Result Discussion

4.5.1 Which type of method performs the best in the empirical experiments?

Overall, hybrid methods generally demonstrate higher accuracy values and balanced accuracies compared to other types of methods for this dataset. Additionally, within the algorithm methods, models processed using SMOTE tend to exhibit superior performance. This combination also falls under the hybrid level. Furthermore, numerous studies indicate a growing trend among researchers to explore hybrid-level methods.

However, there are still some differences in performance metrics among methods that warrant discussion.

4.5.1.1 Higher Balanced Accuracy values in Data-Level methods

We speculate that data-level methods effectively balance the class distribution during data processing, thereby reducing the impact of class imbalance on model training and consequently improving the values of evaluation metrics such as Balanced Accuracy, G-mean score. On the other hand, algorithm methods and hybrid methods rely more on the tuning of the models, which may require more experimentation and adjustments to achieve similar effects.

4.5.1.2 Lower precision and MCC in Hybrid method

In hybrid methods, precision unexpectedly appears to be lower. We speculate that excessive data processing within hybrid methods may lead to overfitting issues. Hybrid methods may not handle data for specific categories effectively, resulting in lower precision compared to data-level or algorithm methods. This could also be attributed to the dataset's characteristics unfavorably affecting the effectiveness of hybrid methods

4.5.1.3 Comparing recall values

In this experiment, data-level methods overall exhibit lower recall values, particularly the KNN classifier. This could be attributed to suboptimal settings of the K value, which have not been fine-tuned to their optimal values. Additionally, it's possible that during oversampling or undersampling, important features were overlooked, resulting in a dataset that is insufficient for better discrimination, leading to lower recall values.

4.5.2 What we found during the experiments?

4.5.2.1 The dataset of financial domain

In numerous financial domains, mixed imbalanced datasets comprising both numerical and categorical attributes are prevalent. For instance, demographic characteristics commonly used in analyses feature attributes like "gender" and "education level," which are categorical, alongside "age," which is numerical. Similarly, in credit risk assessment data, attributes such as "work type" and "property status" are categorical, while "credit amount" and "current amount of all accounts" are numerical. Consequently, classifying mixed imbalanced data entails addressing not only category imbalance but also the continuity of numerical attributes and the discreteness of categorical attributes, presenting a more complex challenge.[30]

4.5.2.2 Domain Knowledge of the dataset

In conducting experiments with MEBoost, although the paper suggests its performance is superior to Adaboost across multiple datasets, no significant results were observed in this experiment. We speculate this discrepancy may be attributed to the necessity for MEBoost to appropriately assign class weights to achieve significant performance, a process requiring a deep understanding of both the dataset and its domain knowledge.

Overall, when using the Adaboost algorithm, it appears that deep research into the dataset is not necessary to achieve good results, representing a more effective approach for dealing with imbalanced datasets.

4.5.2.3 Computation Time

According to study, a dataset consisting at least 100,000 instances can be considered as a big dataset.[38] In this experiment, the Credit card fraud detection dataset used is considered as big dataset. Upon observation, it was noted that some data-level methods incurred significant computational time due to oversampling to address the imbalance in the dataset's minor class. For hybrid methods such as RUSBoost, if not executed using GPU, the processing time exceeds half an hour. Evaluating each method based on computational resources and time, the most optimal method in terms of performance metrics may not necessarily be the best overall method.

4.5.3 Limitation of the empirical experiments

Since the dataset is binary and exhibits an exceptionally high imbalance ratio, many models tend to achieve accuracies close to 1.0. However, testing with a multi-class dataset might showcase different classification results for the models.

5. Conclusion

This survey paper primarily introduces various methods for handling imbalanced datasets and conducts experiments to

showcase the results of each method using performance metrics such as balanced accuracy, precision, recall, AUC, and MCC. We also explore potential reasons for the differences in performance metrics exhibited among the models.

6. REFERENCES

[1] V. S. Spelmen and R. Porkodi, "A Review on Handling Imbalanced Data," in *2018 International Conference on Current Trends towards Converging Technologies (ICCTCT)*, Coimbatore: IEEE, Mar. 2018, pp. 1–11. doi: 10.1109/ICCTCT.2018.8551020.

[2] A. Singh, R. K. Ranjan, and A. Tiwari, "Credit Card Fraud Detection under Extreme Imbalanced Data: A Comparative Study of Data-level Algorithms," *Journal of Experimental & Theoretical Artificial Intelligence*, vol. 34, no. 4, pp. 571–598, Jul. 2022, doi: 10.1080/0952813X.2021.1907795.

[3] J. M. Johnson and T. M. Khoshgoftaar, "Survey on deep learning with class imbalance," *Journal of Big Data*, vol. 6, no. 1, p. 27, Mar. 2019, doi: 10.1186/s40537-019-0192-5.

[4] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic Minority Over-sampling Technique," *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, Jun. 2002, doi: 10.1613/jair.953.

[5] H. He, Y. Bai, E. A. Garcia, and S. Li, "ADASYN: Adaptive synthetic sampling approach for imbalanced learning," in *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, Jun. 2008, pp. 1322–1328. doi: 10.1109/IJCNN.2008.4633969.

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

[7] E. At, A. M, A.-M. F, and S. M, "Classification of Imbalance Data using Tomek Link (T-Link) Combined with Random Under-sampling (RUS) as a Data Reduction Method," *Global J Technol Optim*, vol. 01, no. S1, 2016, doi: 10.4172/2229-8711.S1111.

[8] I. Jamali, M. Bazmara, and S. Jafari, "Feature Selection in Imbalance data sets," vol. 9, no. 3, 2012.

[9] L. Yin, Y. Ge, K. Xiao, X. Wang, and X. Quan, "Feature selection for high-dimensional imbalanced data," *Neurocomputing*, vol. 105, pp. 3–11, Apr. 2013, doi: 10.1016/j.neucom.2012.04.039.

[10] H. He and E. A. Garcia, "Learning from Imbalanced Data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 9, pp. 1263–1284, Sep. 2009, doi: 10.1109/TKDE.2008.239.

[11] N. Thai-Nghe, Z. Gantner, and L. Schmidt-Thieme, "Cost-sensitive learning methods for imbalanced data," in *The 2010 International Joint Conference on Neural Networks (IJCNN)*, Jul. 2010, pp. 1–8. doi: 10.1109/IJCNN.2010.5596486.

[12] J. Tanha, Y. Abdi, N. Samadi, N. Razzaghi, and M. Asadpour, "Boosting methods for multi-class imbalanced data classification: an experimental review," *J Big Data*, vol. 7, no. 1, p. 70, Sep. 2020, doi: 10.1186/s40537-020-00349-y.

[13] A. J. Ferreira and M. A. T. Figueiredo, "Boosting Algorithms: A Review of Methods, Theory, and Applications," in *Ensemble Machine Learning: Methods and Applications*, C. Zhang and Y. Ma, Eds., New York, NY: Springer, 2012, pp. 35–85. doi: 10.1007/978-1-4419-9326-7_2.

[14] L. Breiman, "Bagging predictors," *Mach Learn*, vol. 24, no. 2, pp. 123–140, Aug. 1996, doi: 10.1007/BF00058655.

[15] R. E. Schapire, "A Brief Introduction to Boosting".

[16] M. Galar, A. Fernandez, E. Barrenechea, H. Bustince, and F. Herrera, "A Review on Ensembles for the Class Imbalance Problem: Bagging-, Boosting-, and Hybrid-Based Approaches," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 42, no. 4, pp. 463–484, Jul. 2012, doi: 10.1109/TSMCC.2011.2161285.

[17] Y. Freund and R. E. Schapire, "A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting," *Journal of Computer and System Sciences*, vol. 55, no. 1, pp. 119–139, Aug. 1997, doi: 10.1006/jcss.1997.1504.

[18] G. L. Sahithi, V. Roshmi, Y. V. Sameera, and G. Pradeepini, "Credit Card Fraud Detection using Ensemble Methods in Machine Learning," in *2022 6th International Conference on Trends in Electronics and Informatics (ICOEI)*, Apr. 2022, pp. 1237–1241. doi: 10.1109/ICOEI53556.2022.9776955.

[19] F. Rayhan *et al.*, "MEBoost: Mixing Estimators with Boosting for Imbalanced Data Classification," in *2017 11th International Conference on Software, Knowledge, Information Management and Applications (SKIMA)*, Dec. 2017, pp. 1–6. doi: 10.1109/SKIMA.2017.8294128.

[20] J. Brownlee, "A Gentle Introduction to Threshold-Moving for Imbalanced Classification," MachineLearningMastery.com. Accessed: Mar. 23, 2024. [Online]. Available: https://machinelearningmastery.com/threshold-moving-for-imbalanced-classification/

[21] N. V. Chawla, A. Lazarevic, L. O. Hall, and K. W. Bowyer, "SMOTEBoost: Improving Prediction of the Minority Class in Boosting," in *Knowledge Discovery in Databases: PKDD 2003*, N. Lavrač, D. Gamberger, L. Todorovski, and H. Blockeel, Eds., Berlin, Heidelberg: Springer, 2003, pp. 107–119. doi: 10.1007/978-3-540-39804-2_12.

[22] C. Seiffert, T. M. Khoshgoftaar, J. Van Hulse, and A. Napolitano, "RUSBoost: Improving classification performance when training data is skewed," in *2008 19th International Conference on Pattern Recognition*, Feb. 2008, pp. 1–4. doi: 10.1109/ICPR.2008.4761297.

[23] M. Galar, A. Fernández, E. Barrenechea, and F. Herrera, "EUSBoost: Enhancing ensembles for highly imbalanced data-sets by evolutionary undersampling," *Pattern Recognition*, vol. 46, no. 12, pp. 3460–3471, Dec. 2013, doi: 10.1016/j.patcog.2013.05.006.

[24] J. Gong and H. Kim, "RHSBoost: Improving classification performance in imbalance data," *Computational Statistics & Data Analysis*, vol. 111, pp. 1–13, Jul. 2017, doi: 10.1016/j.csda.2017.01.005.

[25] J. Zhao, J. Jin, S. Chen, R. Zhang, B. Yu, and Q. Liu, "A weighted hybrid ensemble method for classifying imbalanced data," *Knowledge-Based Systems*, vol. 203, p. 106087, Sep. 2020, doi: 10.1016/j.knosys.2020.106087.

[26] L. Prokhorenkova, G. Gusev, A. Vorobev, A. V. Dorogush, and A. Gulin, "CatBoost: unbiased boosting with categorical features," in *Advances in Neural Information Processing Systems*, Curran Associates, Inc., 2018. Accessed: Mar. 10, 2024. [Online]. Available: https://proceedings.neurips.cc/paper/2018/hash/14491b756b3a51daac41c24863285549-Abstract.html

[27] T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, in KDD '16. New York, NY, USA: Association for

Computing Machinery, Aug. 2016, pp. 785–794. doi: 10.1145/2939672.2939785.
[28] D. Veganzones and E. Séverin, "An investigation of bankruptcy prediction in imbalanced datasets," *Decision Support Systems*, vol. 112, pp. 111–124, Aug. 2018, doi: 10.1016/j.dss.2018.06.011.
[29] Y. Sahin, S. Bulkan, and E. Duman, "A cost-sensitive decision tree approach for fraud detection," *Expert Systems with Applications*, vol. 40, no. 15, pp. 5916–5923, Nov. 2013, doi: 10.1016/j.eswa.2013.05.021.
[30] T. Yu and Y. Huo, "Classification of Imbalanced Data Set in Financial Field Based on Combined Algorithm," *Mobile Information Systems*, vol. 2022, p. e1839204, Sep. 2022, doi: 10.1155/2022/1839204.
[31] T. Fawcett, "An introduction to ROC analysis," *Pattern Recognition Letters*, vol. 27, no. 8, pp. 861–874, Jun. 2006, doi: 10.1016/j.patrec.2005.10.010.
[32] T. Saito and M. Rehmsmeier, "The Precision-Recall Plot Is More Informative than the ROC Plot When Evaluating Binary Classifiers on Imbalanced Datasets," *PLOS ONE*, vol. 10, no. 3, p. e0118432, Mar. 2015, doi: 10.1371/journal.pone.0118432.
[33] C. Halimu, A. Kasem, and S. H. S. Newaz, "Empirical Comparison of Area under ROC curve (AUC) and Mathew Correlation Coefficient (MCC) for Evaluating Machine Learning Algorithms on Imbalanced Datasets for Binary Classification," in *Proceedings of the 3rd International Conference on Machine Learning and Soft Computing*, in ICMLSC '19. New York, NY, USA: Association for Computing Machinery, Jan. 2019, pp. 1–6. doi: 10.1145/3310986.3311023.
[34] P. K. Singh, R. Sarkar, and M. Nasipuri, "Significance of non-parametric statistical tests for comparison of classifiers over multiple datasets," *Int. J. Comput. Sci. Math.*, vol. 7, no. 5, pp. 410–442, Jan. 2016, doi: 10.1504/IJCSM.2016.080073.
[35] L. Liu, X. Wu, S. Li, Y. Li, S. Tan, and Y. Bai, "Solving the class imbalance problem using ensemble algorithm: application of screening for aortic dissection," *BMC Medical Informatics and Decision Making*, vol. 22, no. 1, p. 82, Mar. 2022, doi: 10.1186/s12911-022-01821-w.
[36] "Credit Card Fraud Detection." Accessed: Mar. 23, 2024. [Online]. Available: https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud
[37] "A KNN Undersampling Approach for Data Balancing." Accessed: Mar. 21, 2024. [Online]. Available: https://www.scirp.org/journal/paperinformation?paperid=60996
[38] Leevy J. L., Khoshgoftaar T. M., Bauder R. A., and Seliya N., "A survey on addressing high-class imbalance in big data. | Journal of Big Data | EBSCOhost." Accessed: Mar. 23, 2024. [Online]. Available: https://openurl.ebsco.com/contentitem/doi:10.1186%2Fs40537-018-0151-6?sid=ebsco:plink:crawler&id=ebsco:doi:10.1186%2Fs40537-018-0151-6

7. WORK DIVISION

Name	Work description	Contribution
En-Chun, KUO 111423052	Searching hybrid method dealing imbalanced dataset Empirical experiments with Hybrid method Drafting Sections 2,4,5,6	**33%**
Zi-Rong, WANG 112423070	Searching Algorithm-level method dealing imbalanced dataset Empirical experiments with Algorithm-level method Drafting Sections 1,2,3,4	**33%**
Hsuan-Chu, WANG 112423072	Searching Data-level method dealing imbalanced dataset Empirical experiment with Data-level method Drafting Sections 1,2,3,4	**33%**