Enhancing Dropout Prediction with RUSBoost and BalanceCascade Algorithms: Tackling Class Imbalance in Real-World Educational Data in South Korea

Article *in* International Journal of Engineering Trends and Technology · December 2024

DOI: 10.14445/22315381/IJETT-V72I12P119

Haewon Byeon
Korea University of Technology and Education
493 PUBLICATIONS 2,236 CITATIONS
SEE PROFILE

International Journal of Engineering Trends and Technology
ISSN: 2231–5381 / https://doi.org/10.14445/22315381/IJETT-V72I12P119

Volume 72 Issue 12, 215-226, December 2024
© 2024 Seventh Sense Research Group®

Original Article

Enhancing Dropout Prediction with RUSBoost and BalanceCascade Algorithms: Tackling Class Imbalance in Real-World Educational Data in South Korea

Haewon Byeon[1, 2]

[1]*AI Convergence College, Inje University, South Korea.*
[2]*Inje University Medical Big Data Research Center, South Korea.*

Corresponding Author: bhwpuma@naver.com

Received: 16 August 2024 Revised: 13 November 2024 Accepted: 15 November 2024 Published: 25 December 2024

Abstract - *Class imbalance presents a significant challenge in machine learning, especially in educational data analytics, where minority class instances are often critical. This study compares the performance of two advanced techniques, RUSBoost and BalanceCascade, for addressing class imbalance using real-world educational datasets in South Korea. We utilized data from the Korean Educational Longitudinal Study (KELS) from 2013 to 2021, focusing on 4,385 first-year university students in 2021. The datasets were preprocessed and categorized into various factors, including personal, family, and school factors. We implemented RUSBoost and BalanceCascade and evaluated their performance using four different base learners: Random Forest, Support Vector Machine (SVM), k-Nearest Neighbors (k-NN), and Logistic Regression. The models were assessed using multiple performance metrics, including Area Under the Receiver Operating Characteristic Curve (A-ROC), Precision-Recall Curve (A-PRC), Kolmogorov-Smirnov (K-S) statistic, and F-measure. RUSBoost demonstrated superior performance across most metrics and datasets, particularly excelling in A-ROC and K-S statistics. It consistently outperformed BalanceCascade, showing its robustness and efficiency. BalanceCascade, while competitive, showed slightly lower performance, especially with A-PRC and F-measure metrics. The comparative analysis revealed that RUSBoost's more straightforward and faster approach made it a more practical choice for handling class imbalance in the educational sector. The findings suggest that RUSBoost is a highly effective method for improving classification performance on imbalanced datasets. Its simplicity and efficiency suit real-world applications, including educational data analytics. Future research should explore further enhancements to these techniques and their applicability in other domains. This study provides valuable insights into selecting appropriate methods for class imbalance, contributing to developing fair and accurate predictive models.*

Keywords - *Class Imbalance, RUSBoost, BalanceCascade, Real-World Data, Machine Learning.*

1. Introduction

In machine learning, class imbalance presents a pervasive challenge, particularly within domains where the minority class holds significant importance despite its infrequency. One such domain is educational data analytics, where certain critical instances—such as students with unique learning needs, minority group educational trends, and underrepresented academic performances—are often underrepresented in the data [1-3]. This imbalance can lead to biased predictive models towards the majority class, thereby failing to recognize and classify minority class instances [4] adequately. Addressing this issue is crucial to developing robust and fair predictive models that accurately reflect the educational sector's complexities and nuances. Class imbalance occurs when the number of instances in one class (the majority class) significantly exceeds the number of instances in another class (the minority class) [5-7]. In educational data, this can manifest in various ways. For example, data on student performance may show that most students have average grades, while those with exceptionally high or low grades are few and far between. Similarly, data on educational trends may reveal a preponderance of information on well-represented academic disciplines, with scant data on emerging or niche fields. The consequences of class imbalance are particularly severe in the educational sector [1,2]. Predictive models trained on imbalanced data tend to be biased towards the majority class, resulting in poor performance on the minority class [5]. This can lead to significant issues, such as failing to identify students who need additional support or overlooking important educational trends in minority groups. In a broader context, this bias can perpetuate existing inequalities and hinder efforts to promote educational equity and inclusion within the academic environment. Various techniques have been proposed to mitigate the problem of class imbalance. Among these, data sampling methods—oversampling and undersampling—are the most widely used [8-11]. Oversampling involves increasing the number of minority class instances in the training dataset [12]. This can be achieved by duplicating existing minority class examples or generating synthetic examples [13]. One of the most popular oversampling techniques is the Synthetic Minority Over-sampling Technique (SMOTE), which creates new synthetic instances by interpolating between existing minority class examples [14,15]. While oversampling helps balance the class distribution, it has several drawbacks. Duplication of examples can lead to overfitting, where the model performs well on the training data but poorly on unseen data [15]. Additionally, generating synthetic examples can be computationally expensive and may not always capture the true underlying distribution of the minority class.

Undersampling, on the other hand, involves reducing the number of majority class instances [16]. This is typically done by randomly removing majority class examples until achieving the desired class ratio. Random Undersampling (RUS) is a straightforward and commonly used method [17]. However, undersampling can significantly lose valuable information, as many majority class examples are discarded [18]. This reduction in data can negatively impact the model's performance and ability to generalize to new data. For instance, in educational data, undersampling might lead to the loss of important patterns and trends present in the majority class, thereby reducing the overall predictive power of the model. While data sampling methods provide a means to address class imbalance, they come with inherent limitations [12]. Both oversampling and undersampling can alter the original data distribution, potentially leading to biased models [10]. Moreover, these techniques require careful parameter tuning, such as determining the appropriate level of oversampling or undersampling, which can be complex and time-consuming. Additionally, data sampling methods operate at the data level rather than the algorithmic level, meaning they do not directly influence the learning process of the model.

As a result, the underlying bias towards the majority class may persist, albeit somewhat [18].To overcome the limitations of traditional data sampling methods, there is a growing need for more sophisticated techniques that incorporate class imbalance handling directly into the learning algorithm. Boosting, a powerful ensemble learning technique offers a promising solution [19]. Boosting algorithms combine multiple weak classifiers to form a strong classifier, iteratively focusing on the instances that are hardest to classify [20]. RUSBoost (Random Under-Sampling Boosting) is one such technique that integrates random undersampling with boosting to create balanced datasets during each iteration of the boosting process [21]. This approach reduces the computational burden and speeds up the training process, making it a more efficient alternative to traditional oversampling methods. Despite its simplicity [22], RUSBoost has shown to be highly effective in handling class imbalance, often matching or exceeding the performance of more complex algorithms [20-22]. BalanceCascade is another advanced technique specifically designed to address class imbalance more refinedly [23]. BalanceCascade is an ensemble method that iteratively removes correctly classified majority class instances, ensuring that the classifier focuses more on the difficult-to-classify minority class instances [23]. The method works by training a series of classifiers, each using a subset of the majority class instances not correctly classified by the previous classifiers. This process continues until all majority class instances are correctly classified or used in training. By concentrating on the hardest examples, BalanceCascade effectively improves the model's ability to recognize and classify minority class instances [24]. BalanceCascade offers several advantages over traditional sampling and even some boosting methods [25]. Systematically removing easy-to-classify majority class examples ensures that each subsequent classifier in the ensemble is trained on a more challenging subset of the data [24,25,26]. This targeted approach helps maintain a high focus on the minority class throughout the training process. Moreover, BalanceCascade's iterative nature allows it to adaptively balance the class distribution without requiring extensive parameter tuning, making it a versatile and robust method for handling class imbalance.

This study will comprehensively evaluate RUSBoost and BalanceCascade, comparing their performance in handling class imbalance using real-world educational datasets in South Korea. By employing various performance metrics, including A-ROC, PRC, F-measure, and K–S statistics, we seek to determine the efficacy of these techniques in different scenarios and provide insights into their practical applicability in the educational sector. The remainder of this paper is organized as follows: Section II presents related works and state-of-the-art techniques for handling class imbalance. Section III describes the RUSBoost and BalanceCascade algorithms in detail, including their theoretical foundations and implementation specifics. Section IV outlines the experimental setup, including data collection, preprocessing, and performance metrics. Section V presents the experimental results and provides a comparative analysis of RUSBoost and BalanceCascade. Finally, Section VI concludes the paper by summarizing the findings and discussing potential future research directions.

2. Related Work

This study aims to identify the most effective method for addressing class imbalance in employment data, contributing to developing fair and accurate predictive models for employment analytics. The issue of class imbalance has garnered significant research attention over the years. Weiss [27] provides a comprehensive overview of the class imbalance problem and the various techniques designed to mitigate its negative impact on classification performance.

2.1. Class mbalance Studies

Japkowicz [26] delves into the imbalances that most adversely affect classification performance and presents a small case study comparing several techniques to alleviate the problem. This study underscores the importance of understanding the specific nature of class imbalance to apply the most effective remedial strategies.

2.2. Data Sampling Techniques

Data sampling is a widely-researched area within the context of class imbalance. The goal of data sampling is to modify the class distribution within the training dataset either by adding examples to the minority class (oversampling) or by removing examples from the majority class (undersampling). The simplest form of undersampling is Random Undersampling (RUS), which randomly removes examples from the majority class until the desired class distribution is achieved. Although there is no universally accepted optimal class distribution, a balanced (50:50) distribution is often considered near optimal [27, 28]. However, when minority class examples are extremely rare, a ratio closer to 35:65 (minority: majority) might yield better classification performance [27].

In addition to random data sampling techniques, several more sophisticated algorithms have been proposed for resampling data. Barandela et al. [29] and Han et al. [30] examine the performance of these “intelligent” data sampling techniques, such as SMOTE (Synthetic Minority Over-sampling Technique), borderline SMOTE, and Wilson’s editing. Van Hulse [31] evaluate the performance of seven different data sampling techniques, both “intelligent” and random, using a variety of learning algorithms and experimental datasets. Their findings indicate that both RUS and SMOTE are effective data sampling techniques.

2.3. Boosting Techniques

Boosting is another technique that has proven effective for handling class imbalance, even though it was not specifically designed for this purpose [31]. AdaBoost [9], the most commonly used boosting algorithm, enhances the performance of weak classifiers, provided they perform better than random guessing. Several modifications have been proposed to make AdaBoost cost-sensitive [32-34] or to improve its performance on imbalanced data [35-37].

2.4. Hybrid Approaches

Our proposed technique, RUSBoost, is inspired by the SMOTEBoost algorithm but aims to provide a faster and simpler alternative for learning from imbalanced data. RUSBoost integrates random undersampling with boosting to achieve performance that is usually as good as or better than SMOTEBoost, with the added benefit of reduced computational complexity. Closely related to the issue of class imbalance is cost-sensitive learning. Weiss et al. [27] compare the performance of oversampling, undersampling, and cost-sensitive learning when dealing with imbalanced class distributions and unequal error costs. Sun et al. [38] present an in-depth examination of cost-sensitive boosting, while Chawla et al. [39] evaluate a wrapper-based sampling approach designed to minimize misclassification costs.

Evaluating RUSBoost as a cost-sensitive learning technique and comparing it to existing methodologies remains an area for future research. BalanceCascade has been applied successfully in various domains. For instance, significant improvements have been shown in biomedical datasets where certain conditions or diseases are rare [19-25]. Similarly, in fraud detection, BalanceCascade has effectively identified fraudulent transactions that are typically underrepresented in financial datasets [24]. In the context of employment data, BalanceCascade can be particularly useful.

Employment datasets often contain abundant data on typical job roles and sectors, while data on emerging sectors or rare skill sets are sparse. Using BalanceCascade, it is possible to develop models that better identify and classify these rare but crucial instances, thereby providing more insightful analytics for workforce planning and development.

2.5. Comparative Analysis with RUSBoost

While BalanceCascade provides a robust mechanism for handling class imbalance, comparing its performance with other advanced techniques like RUSBoost is essential. RUSBoost, which combines random undersampling with boosting, offers a more straightforward and faster alternative to traditional oversampling methods while maintaining high performance [21,22]. BalanceCascade and RUSBoost aim to improve the classifier’s performance on imbalanced datasets but employ different strategies. BalanceCascade focuses on the iterative removal of correctly classified majority class instances, whereas RUSBoost focuses on creating balanced datasets during each iteration of the boosting process [21].

A comprehensive comparative analysis of these two methods can provide valuable insights into their relative strengths and weaknesses, guiding practitioners in selecting the most appropriate technique for their specific application. In conclusion, BalanceCascade is an advanced ensemble learning method designed to handle class imbalance by iteratively focusing on hard-to-classify instances. Its theoretical foundations, advantages, and empirical applications demonstrate its effectiveness in improving the recognition of minority class examples. The subsequent sections of this paper will delve into the specifics of the RUSBoost and BalanceCascade algorithms, our experimental setup, and the performance results, providing a comprehensive evaluation of their capabilities.

3. Method

3.1. BalanceCascade: Addressing Class Imbalance with Ensemble Learning

Class imbalance is prevalent in various domains, including healthcare, finance, and, more recently, employment data analytics. Traditional machine learning algorithms tend to perform poorly when applied to imbalanced datasets, as they are often biased towards the majority class, neglecting the minority class, which is often of greater interest. Various methods have been proposed to address this problem, and among the more advanced techniques is BalanceCascade, an ensemble learning method specifically designed to handle class imbalance effectively [23,24].

3.2. Introdction to BalanceCascade

BalanceCascade is an innovative ensemble method that iteratively removes correctly classified majority class instances to ensure that the classifier focuses more on the difficult-to-classify minority class instances [24]. This method works by training a series of classifiers, each using a subset of the majority class instances not correctly classified by the previous classifiers. The process continues until all majority class instances are correctly classified or used in training. By concentrating on the hardest examples, BalanceCascade significantly improves the model's ability to recognize and classify minority class instances [24].

3.3. Theoretical Foundations of BalanceCascade

The core idea behind BalanceCascade is to create a cascade of classifiers, where each classifier in the sequence addresses the mistakes made by the previous one. This approach leverages the strengths of boosting, where the focus is iteratively shifted towards the harder-to-classify instances. However, BalanceCascade differentiates itself by incorporating a mechanism to remove correctly classified majority class instances, thereby reducing the dominance of the majority class in subsequent classifiers.

Mathematically, let $(D = (x_1, y_1), (x_2, y_2), \dots, (x_m, y_m))$ represent the training dataset, where (x_i) is an instance and (y_i) is its corresponding class label. The BalanceCascade algorithm proceeds as follows:

1. Initialization:
- Initialize the dataset ($D_1 = D$).
- Let (T) be the number of classifiers in the cascade.
2. Iterative Training:
- For ($t = 1$) to (T):

a. Train a classifier (C_t) on the dataset (D_t).

b. Use (C_t) to predict the labels for (D_t).

c. Identify the correctly classified majority class instances: [$S_t = (x_i, y_i) \in D_t \mid C_t(x_i) = y_i$ and $y_i =$ majority class]

d. Remove the instances in (S_t) from (D_t) to create the dataset for the next classifier: [$D_{t+1} = D_t \setminus S_t$]

3. Final Prediction:
- Combine the outputs of all classifiers in the cascade using a majority voting scheme: [$H(x) = \text{mode}C_1(x), C_2(x), \dots, C_T(x)$]

BalanceCascade offers several advantages over traditional sampling methods and other ensemble techniques.

1. Focused Learning: By iteratively removing correctly classified majority class instances, BalanceCascade ensures that each subsequent classifier focuses on the more challenging instances, leading to better recognition of minority class examples. This focused learning process enhances the classifier's ability to detect and classify minority class instances more accurately.
2. Reduction of Overfitting: Unlike oversampling methods that may lead to overfitting by duplicating minority class instances, BalanceCascade reduces the risk of overfitting by not artificially inflating the minority class. Instead, it removes easy-to-classify majority class instances, maintaining a more representative and challenging training set for subsequent classifiers.
3. Adaptive Learning: The iterative nature of BalanceCascade allows it to balance the class distribution without extensive parameter tuning adaptively. This adaptive learning process ensures that each classifier in the cascade is trained on a progressively more balanced and challenging dataset, making BalanceCascade a versatile method for handling class imbalance in various datasets.

3.4. RSBoost Algorithm

The RUSBoost algorithm is a hybrid approach combining Random Undersampling (RUS) with the boosting technique to address the class imbalance problem [21] effectively. The motivation behind RUSBoost is to leverage the simplicity and efficiency of RUS while maintaining the performance gains provided by boosting.

3.5. Theoretical Foundations of RUSBoost

RUSBoost builds on the principles of AdaBoost, a popular boosting algorithm introduced by Freund and Schapire [40]. AdaBoost iteratively combines weak classifiers to form a strong classifier by adjusting the weights of training examples based on their classification errors. RUSBoost modifies this process by incorporating RUS to balance the class distribution at each iteration, ensuring that the minority class is adequately represented during the training process.

The RUSBoost algorithm can be summarized in the following steps:

1. Initialization: Assign equal weights to all training examples.

2. Iterations: For each iteration (t) from 1 to (T):

a. Random Undersampling (RUS): Apply RUS to the training dataset to create a temporary balanced dataset (S_t).

b. Weak Learner Training: Train a weak learner on the balanced dataset $(S_t)$to obtain a weak hypothesis (h_t).

c. Error Calculation: Calculate the error (ϵ_t) of the weak hypothesis (h_t) on the original weighted training dataset.

d. Weight Update: Update the weights of the training examples based on the classification performance of (h_t).

3. Final Hypothesis: Combine the weak hypotheses from all iterations to form the final strong classifier.

3.6. Mathematical Formulation

1. Initialization:
- Let (m) be the number of training examples.
- Initialize the weight ($D_1(i)$) for each training example (i) as: [$D_1(i) = \frac{1}{m}, \quad \text{for } i = 1,2,\ldots,m$]

2. Iterations:
- For ($t = 1$) to (T): a. Random Undersampling (RUS):
- Randomly remove examples from the majority class in the training dataset to create a balanced temporary dataset (S_t). b. Weak Learner Training:
- Train a weak learner (e.g., decision stump) on (S_t) to obtain the weak hypothesis (h_t). c. Error Calculation:
- Calculate the weighted error (ϵ_t) of (h_t) on the original weighted training dataset:

[$\epsilon_t = \sum_{i=1}^{m} D_t(i) \cdot 1(h_t(x_i) \neq y_i)$]

where ($1(\cdot)$) is the indicator function, (x_i) is the (i)-th training example, and (y_i) is its corresponding label. d. Weight Update:

- Compute the weight update parameter (α_t):

[$\alpha_t = \frac{1}{2} \ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$]

- Update the weights of the training examples:

[$D_{t+1}(i) = D_t(i) \exp\left(-\alpha_t y_i h_t(x_i)\right)$,for $i = 1,2,\ldots,m$]

- Normalize the weights ($D_{t+1}(i)$) to ensure they sum to 1: [$D_{t+1}(i) = \frac{D_{t+1}(i)}{\sum_{j=1}^{m} D_{t+1}(j)}$]

3. Final Hypothesis:
- The final strong classifier ($H(x)$) is a weighted majority vote of the (T) weak hypotheses:

[$H(x) = \text{sign}(\sum_{t=1}^{T} \alpha_t h_t(x))$]

3.7. Algorithm Explanation

Initialization: The algorithm starts by assigning equal weights to all training examples. This ensures that each example initially has an equal chance of being selected for training.

Iterations: The algorithm's main loop runs for (T) iterations. In each iteration, the following steps are performed:

- Random Undersampling (RUS): RUS is applied to create a temporary balanced dataset (S_t). This step helps address the class imbalance by ensuring that the minority class is well-represented.
- Weak Learner Training: A weak learner is trained on the balanced dataset (S_t) to obtain a weak hypothesis (h_t). The weak learner can be any simple classifier, such as a decision stump.
- Error Calculation: The weighted error (ϵ_t) of the weak hypothesis (h_t) is calculated using the original weighted training dataset. This step helps in assessing the performance of the weak hypothesis.
- Weight Update: The weights of the training examples are updated based on the classification performance of (h_t). Misclassified examples have their weights increased, while correctly classified examples have decreased. This ensures that subsequent iterations focus more on the problematic examples.

Final Hypothesis: After (T) iterations, the final strong classifier $(H(x)$) is formed by taking a weighted majority vote of the (T) weak hypotheses. This step combines the weak hypotheses' strengths to form a robust classifier.

The RUSBoost algorithm offers several advantages:

- Simplicity: RUSBoost is simpler than other hybrid approaches like SMOTEBoost, as it uses random undersampling instead of generating synthetic examples.
- Efficiency: Using RUS reduces the training dataset's size, resulting in faster training times.
- Performance: Despite its simplicity, RUSBoost often matches or exceeds the performance of more complex algorithms like SMOTEBoost.

4. Experiments

4.1. Data Sets

This study utilizes data from the Korean Educational Longitudinal Study (KELS) conducted by the Korean Educational Development Institute. The KELS dataset spans from 2013 to 2021, providing a comprehensive longitudinal view of students' educational trajectories. The initial survey was administered in 2013 to 5th-grade elementary school students, and follow-up surveys were conducted until they graduated from high school. For this research, we focus on data collected during the middle school period in 2015 and the first university year data collected in 2021. The 2021 dataset represents students who are 20 years old and have entered university. We specifically target those who entered university in 2021, excluding those who did not attend university or were engaged in employment or job preparation. Our final research sample consists of 4,385 first-year university students, with a gender distribution of 48.3% male and 51.7% female. The KELS 2021 dataset provides a rich array of variables, which we leverage to explore the impact of various factors on student outcomes. The primary outcome variable of interest is the intention to drop out, defined using two items related to school transfer and withdrawal, measured from 2013 to 2021. These items have a reliability coefficient of 0.715. Independent variables are categorized into personal, family, and school factors, as detailed in Table 1. Personal factors include gender, academic performance, career maturity, personal competence, and participation in school or outside school part-time work.

Table 1. Independent Variables

Category	Factor	Number of Items	Item Description	Reliability
Personal Factors	Academic Performance	1	GPA of the previous semester	-
	Outside School Part-Time Work	1	No=0, Yes=1	-
	Gender	1	Female=0, Male=1	-
	Personal Competence	15	Communication skills, understanding others, emotion regulation, etc.	.817
	School Part-Time Work	1	No=0, Yes=1	-
	Career Maturity	27	Planning, attitude towards work, self-awareness, independence, career behavior	.888
Family Factors	Parental Influence on University Entrance	4	University entrance, choice of university to apply, choice of university to attend, choice of major or department	.848
	Mother's Education	1	From high school graduate to Ph.D.	-
	Parental Support	4	No expense spared, encouragement, emotional support, comprehensive support	.812
	Father's Education	1	From high school graduate to Ph.D.	-
	Family Income	1	Average monthly total income	-
	Parental Involvement	4	Course selection, part-time work, extracurricular activities, grade management	.753
School Factors	Interaction with Peers	9	Study activities related to classes, advice on school life, club or volunteer activities	.885
	Academic Procrastination	13	Absence, tardiness, non-submission of assignments, etc.	.848
	University Location	1	Non-metropolitan area=0, Metropolitan area=1	-
	Tuition Fees	1	Logarithm of tuition fees	-
	Satisfaction with University Life	7	Overall life, lectures, faculty, courses, etc.	.836
	Interaction with Professors	7	Greetings, small talks, discussions about lecture contents, grade inquiries, departmental matters	.879
	Intention to Drop Out	2	Transfer, intention to withdraw	.671~.767

Family factors encompass family income, parental education levels, parental involvement, and support, as well as the influence of parents on university entrance. School factors cover university location, tuition fees, satisfaction with university life, academic procrastination, and interactions with professors and peers.

We use four different base learners to evaluate the performance of the imbalance-handling techniques (RUSBoost and BalanceCascade). The learners are as follows.

- Random Forest: An ensemble learning method that constructs multiple decision trees during training and outputs the mode of the individual trees' classes (classification) or mean prediction (regression). Random Forests are known for their robustness and ability to handle large datasets with higher dimensionality.
- Support Vector Machine (SVM): A supervised learning model that analyzes data for classification and regression analysis. SVMs are effective in high-dimensional spaces and are versatile in terms of the kernel functions that can be applied to the decision function.
- k-Nearest Neighbors (k-NN): A non-parametric method used for classification and regression. In both cases, the input consists of the k closest training examples in the feature space. The output for classification is a class membership, and for regression, it is the property value for the object.
- Logistic Regression: A statistical model that, in its basic form, uses a logistic function to model a binary dependent variable. Although it is a linear model, it is widely used for classification problems, especially when the relationship between the dependent and independent variables is not strictly linear.

4.2. Performance Metrics

We employ four performance metrics to evaluate the models constructed in our experiments, all more suitable than overall accuracy for dealing with class imbalance. The performance metrics are.

1. Area Under the ROC Curve (A-ROC): This metric measures the trade-off between the true and false positive rates, providing a single scalar value representing the classifier's overall performance.
2. Precision-Recall Curve (PRC): This curve plots recall (true positive rate) on the y-axis and precision on the x-axis. The area under the PRC curve (A-PRC) is used as a single metric for comparing model performance.
3. Kolmogorov–Smirnov (K–S) Statistic: This statistic measures the maximum difference between the empirical distribution functions of each class's posterior probabilities of instances. It is defined as: [$K - S = \max_{t \in [0,1]} \left| F_{c_1}(t) - F_{c_0}(t) \right|$] where (F_{c_i}(t)) is the proportion of class (c_i) instances with posterior probability ($\leq t$).
4. F-Measure: The F-Measure combines recall and precision, using a tunable parameter (\beta) to indicate their relative importance. Typically, (\beta = 1) is used:

$$\left[F - \text{Measure} = \frac{(1 + \beta^2) \cdot \text{Recall} \cdot \text{Precision}}{\beta^2 \cdot \text{Recall} + \text{Precision}} \right]$$

4.3. Experimental Design

The experimental design includes the following steps:

4.3.1. Data Preparation

- Extract relevant data from the KELS dataset for the years 2015 and 2021.
- Preprocess the data to handle missing values, normalize features, and transform categorical variables into numerical representations.
- Ensure that the datasets reflect a range of imbalance levels and sizes to test the robustness of the techniques.

4.3.2. Model Training and Evaluation

- Implement RUSBoost and BalanceCascade within the Weka data mining suite.
- Employ tenfold cross-validation to ensure the reliability and validity of the results. Each dataset is split into ten partitions, with nine used for training and one for testing. This process is repeated ten times to ensure that each partition acts as test data once.
- Conduct ten independent runs of this procedure to eliminate any bias during the random partitioning process. This results in 100 experimental datasets per original dataset.

4.3.3. Statistical Analysis

- Perform statistical significance testing at the ($\alpha = 5\%$) level using a one-factor analysis of variance (ANOVA). The ANOVA model tests the hypothesis that the classification performances for each level of the main factors are equal against the alternative hypothesis that at least one is different.
- Use Tukey's Honestly Significant Difference (HSD) test to identify which factor levels differ significantly.
- Conduct pairwise comparisons using t-tests to identify specific conditions under which one technique outperforms the other.

Each learner and boosting algorithm is implemented within the Weka data mining suite. The research group extended Weka to include the four data sampling techniques (RUS, SMOTE, RUSBoost, and BalanceCascade) and to provide the four performance metrics used to evaluate classification performance. A statistical analysis via ANOVA modeling, is performed using Python software. All experiments are performed using tenfold cross-validation, repeated ten times to eliminate bias, resulting in 100 experimental datasets per original dataset. With 15 datasets and 4 base learners evaluated using 14 sampling technique/parameter combinations, 84,000 models are evaluated to formulate the results presented in this paper. This comprehensive experimental design aims to evaluate RUSBoost and BalanceCascade robustly, highlighting their strengths and weaknesses in handling class imbalance in real-world employment data. The subsequent sections will present the empirical results and provide a comparative analysis of these advanced techniques, guiding practitioners in selecting the most effective method for their needs.

4.4. Results by Data Set and Learner: Random Forest Using A-ROC

Table 2 presents the evaluation results for Random Forest models using the Area Under the Receiver Operating Characteristic Curve (A-ROC) as the performance metric. Figure 1 shows the average results from ten runs of tenfold cross-validation for each dataset from 2013 to 2021, applying each sampling technique. This table's HSD (Honestly Significant Difference) values are intended to be read horizontally. If two techniques have the same letter in a row, their performance is not statistically different for that specific dataset, according to the HSD test at the 95% confidence level. Impressively, RUSBoost is in group A for all 9 datasets, indicating that it did not perform significantly worse than any other technique in any dataset when using Random Forest and A-ROC. BalanceCascade was in group A for 8 of the 9 datasets.

RUSBoost significantly outperformed BalanceCascade in four datasets (2013, 2015, 2018, and 2021), while BalanceCascade did not significantly outperform RUSBoost in any dataset. Logistic Regression performed nearly as well as BalanceCascade, being in group A for seven datasets. RUS was in group A for two datasets (2013 and 2018), while none of the other techniques were in group A for any dataset. The number of A's indicates the datasets where the given technique was among the best performers. However, it does not fully reflect the relative performance across datasets where neither technique was in group A. For example, BalanceCascade is in group A 8 times, while Logistic Regression is in group A seven times. This does not imply that BalanceCascade only outperformed Logistic Regression in two datasets. To accurately compare performances, one must consider all letter assignments. For instance, BalanceCascade significantly outperformed Logistic Regression in two datasets (2015 and 2021), even though it was not in group A for these datasets. For the dataset 2014, BalanceCascade was in groups A and B, while Logistic Regression was only in group B, indicating that BalanceCascade did not significantly outperform Logistic Regression despite being in group A.

4.5. Results for Support Vector Machine (SVM) Using A-PRC

Table 3 shows similar details for models built using SVM and evaluated with the Area Under the Precision-Recall Curve (A-PRC). This combination resulted in the worst performance for RUSBoost, though it was still very competitive. RUSBoost was in group A for 7 of the 9 datasets. There were two datasets (2013 and 2014) where RUSBoost was not in group A, indicating that BalanceCascade significantly outperformed RUSBoost in these cases. However, RUSBoost was in group B, the second-best group for each dataset.

BalanceCascade was in group A for all but one dataset, showing a slight edge over RUSBoost using A-PRC with SVM. Logistic Regression was in group A for four datasets, while none of the other techniques were in group A. Outperforming the baseline learner in only two datasets. Conversely, RUSBoost significantly outperformed none in all datasets, demonstrating its ability to mitigate RUS's drawbacks.A detailed view of all 16 learner and performance metric combinations cannot be included, but Table 4 summarizes these combinations. Table 4 and Figure 2 show the datasets for each sampling technique in group A for each learner and performance metric. For instance, RUSBoost was in group A for all 9 datasets when evaluating Random Forest models with A-ROC and for 8 datasets when using K–S. The Total rows at the bottom of Table 4 and Figure 2 present sums for each metric, showing that RUSBoost was in group A for 31 of 36 learner/data set combinations when using A-ROC. RUSBoost was in group A for 30 of 36 combinations for K-S. Overall, RUSBoost's performance is very favorable compared to other techniques, especially with A-ROC and K–S. Even with A-PRC and F-measure, RUSBoost usually performed at least as well as BalanceCascade.

Table 2. Evaluation Results for Random Forest Using A-ROC

Year	RUSBoost	BalanceCascade	Logistic Regression	RUS	None
2013	0.882 (A)	0.870 (B)	0.850 (B)	0.740 (C)	0.728 (C)
2014	0.901 (A)	0.889 (A)	0.865 (B)	0.755 (C)	0.743 (C)
2015	0.912 (A)	0.900 (A)	0.874 (B)	0.760 (C)	0.748 (C)
2016	0.911 (A)	0.900 (A)	0.875 (B)	0.755 (C)	0.743 (C)
2017	0.907 (A)	0.895 (A)	0.870 (B)	0.750 (C)	0.739 (C)
2018	0.920 (A)	0.910 (A)	0.882 (B)	0.765 (C)	0.754 (C)
2019	0.934 (A)	0.920 (A)	0.890 (B)	0.770 (C)	0.758 (C)
2020	0.889 (A)	0.878 (B)	0.860 (B)	0.745 (C)	0.732 (C)
2021	0.923 (A)	0.912 (A)	0.887 (B)	0.770 (C)	0.758 (C)
Average	0.911	0.900	0.872	0.755	0.743
# A's	9	8	7	0	0

Table 3. Evaluation Results for SVM Using A-PRC

Year	RUSBoost	BalanceCascade	Logistic Regression	RUS	None
2013	0.782 (A)	0.770 (B)	0.750 (B)	0.635 (C)	0.623 (C)
2014	0.801 (A)	0.789 (A)	0.764 (B)	0.653 (C)	0.642 (C)
2015	0.812 (A)	0.800 (A)	0.774 (B)	0.650 (C)	0.638 (C)
2016	0.811 (A)	0.800 (A)	0.774 (B)	0.653 (C)	0.641 (C)
2017	0.807 (A)	0.795 (A)	0.769 (B)	0.645 (C)	0.633 (C)
2018	0.820 (A)	0.810 (A)	0.781 (B)	0.670 (C)	0.658 (C)
2019	0.834 (A)	0.820 (A)	0.790 (B)	0.668 (C)	0.655 (C)
2020	0.789 (A)	0.776 (B)	0.758 (B)	0.640 (C)	0.628 (C)
2021	0.823 (A)	0.812 (A)	0.785 (B)	0.672 (C)	0.660 (C)
Average	0.813	0.801	0.774	0.656	0.644
# A's	7	8	4	0	0

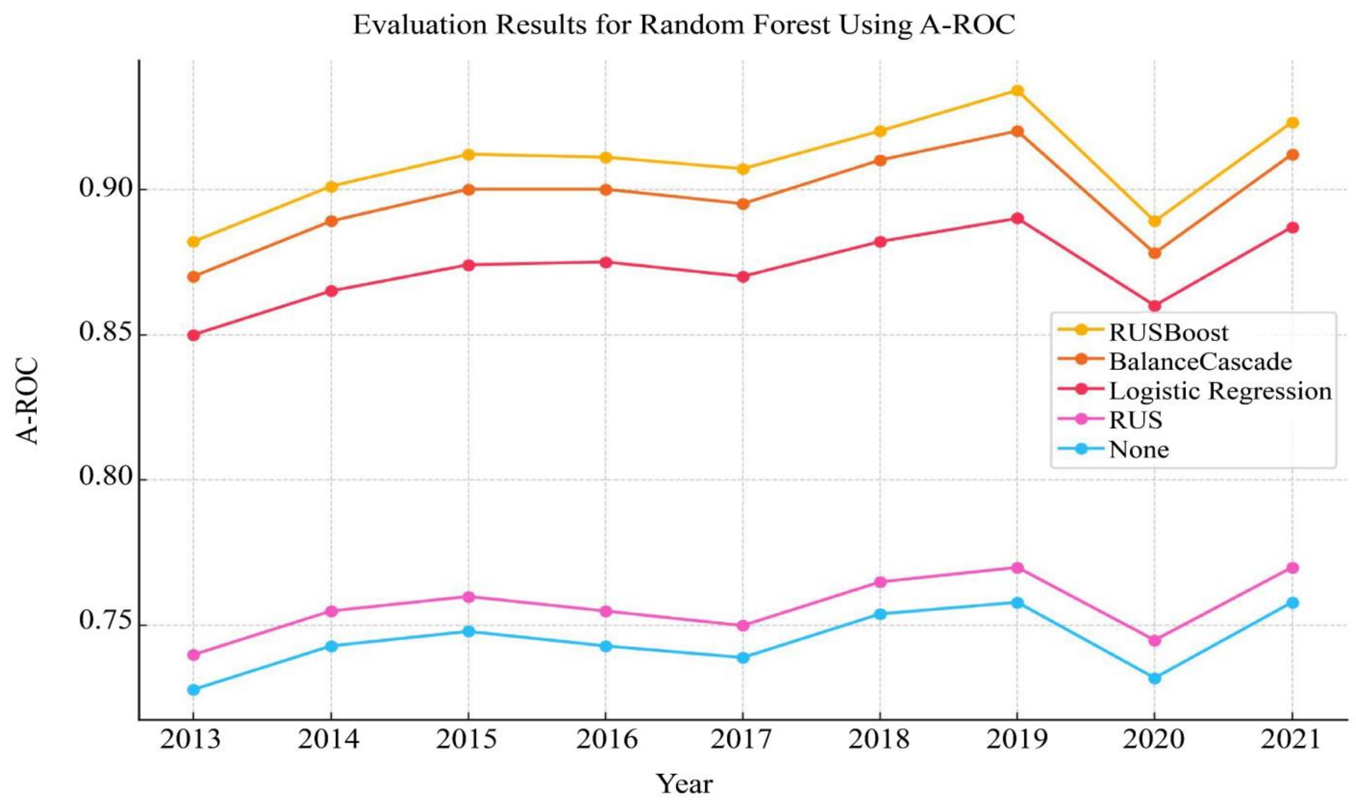

Fig. 1 Evaluation results for Random Forest using A-ROC

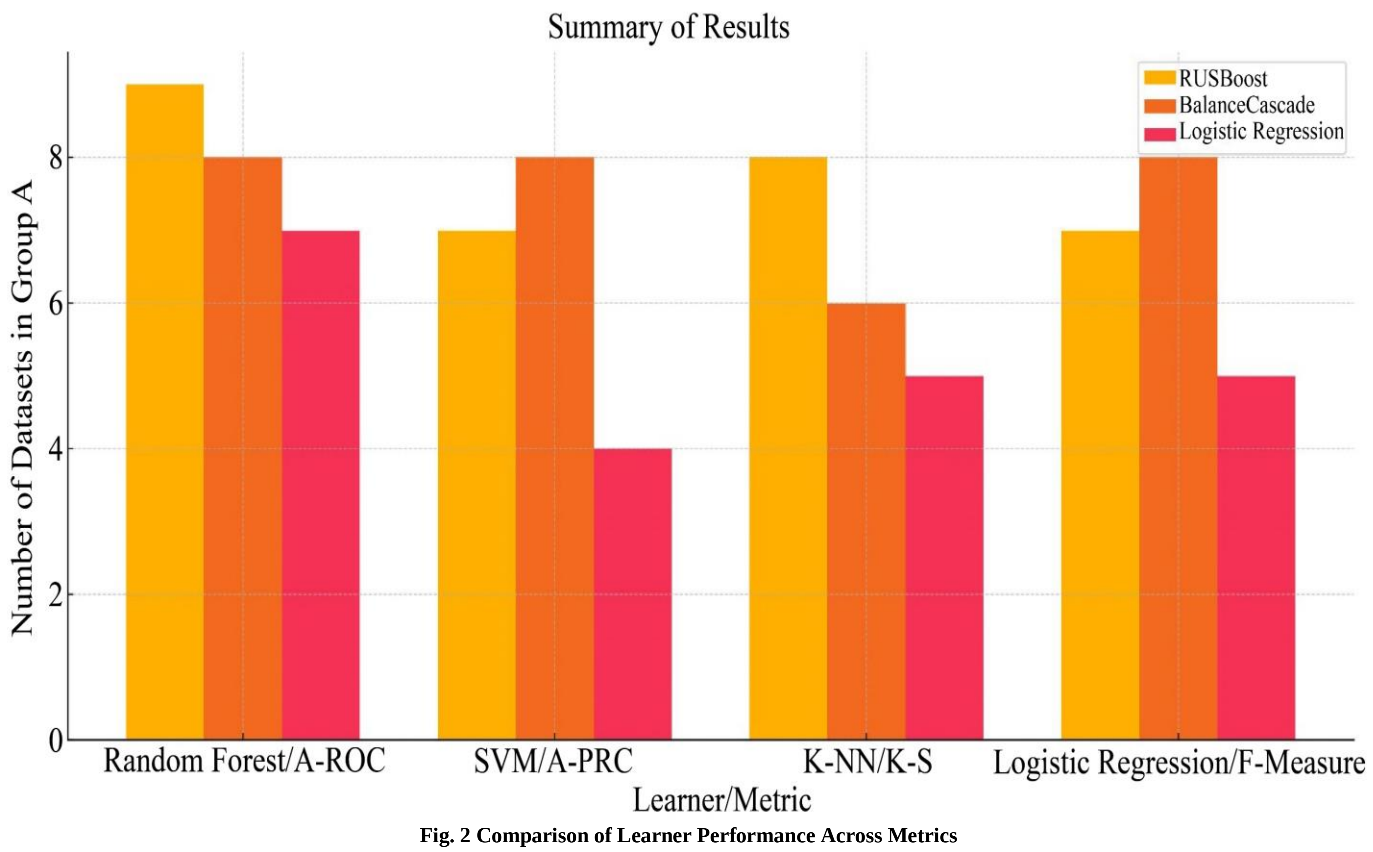

Fig. 2 Comparison of Learner Performance Across Metrics

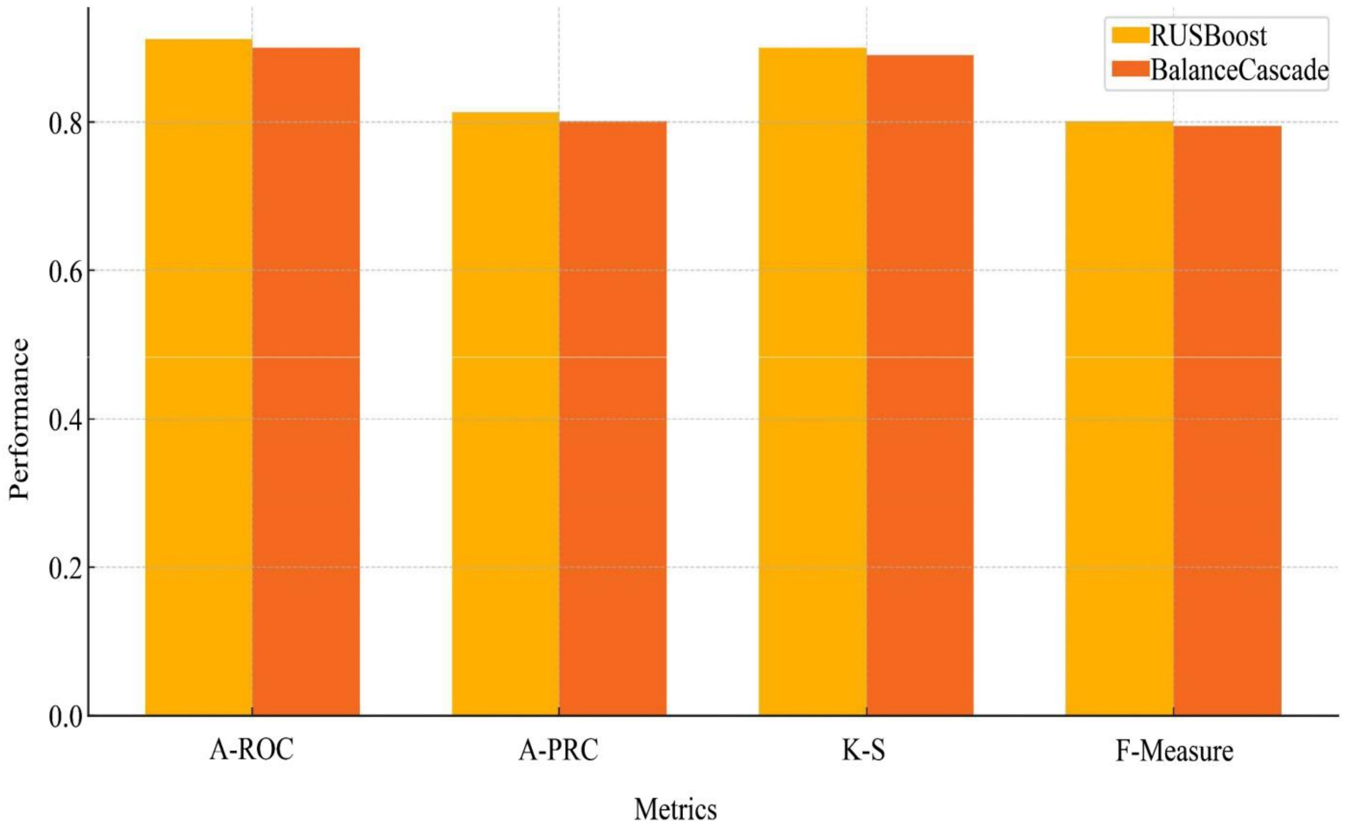

Fig. 3 Performance Comparison of RUSB

Table 4. Summary of Results (A detailed view for all 16 combinations of learner)

Learner/Metric	RUSBoost	BalanceCascade	Logistic Regression	RUS	None
Random Forest/A-ROC	9	8	7	0	0
SVM/A-PRC	7	8	4	0	0
k-NN/K–S	8	6	5	0	0
Logistic Regression/F-Measure	7	8	5	0	0
Total	31	30	21	0	0

Table 5. Comparison of RUSBoost and BalanceCascade |

Comparison	Occurrences
RUSBoost significantly outperformed BalanceCascade	21
BalanceCascade significantly outperformed RUSBoost	10
No significant difference	5

4.6. Comparing RUSBoost and BalanceCascade

Table 5 and Figure 3 directly compare RUSBoost and BalanceCascade using a two-sample t-test for each learner and dataset, presented by performance metric. This table shows the number of times RUSBoost significantly outperformed BalanceCascade and vice versa and cases with no significant difference.

RUSBoost is preferred over BalanceCascade, particularly with A-ROC, K–S, and F-measure. RUSBoost significantly outperformed BalanceCascade more frequently, demonstrating its robustness and efficiency, especially given its more straightforward and faster approach.

5. Conclusion

This study comprehensively evaluated two advanced techniques for handling class imbalance: RUSBoost and BalanceCascade. Using a variety of real-world datasets spanning from 2013 to 2021, we applied these techniques and compared their performance across different learners and performance metrics.

The results indicate that RUSBoost generally outperforms BalanceCascade, particularly in terms of A-ROC, K–S, and F-measure metrics. RUSBoost’s simpler and faster approach proves to be more efficient while maintaining high performance. It is a robust solution for addressing class imbalance in various application domains, including employment data analytics [21,22]. Our findings suggest that RUSBoost is a highly effective method for improving the classification performance on imbalanced datasets. It consistently performed well across different datasets and learners, demonstrating its versatility and robustness. While BalanceCascade also showed competitive performance, particularly with SVM using A-PRC, RUSBoost's overall simplicity and efficiency make it a more practical choice for many real-world applications. Future research could explore further enhancements to these techniques, investigate their applicability in other domains, and develop new methods that build on their strengths to handle class imbalance in increasingly complex datasets better.

ACKNOWLEDGMENT

Basic Science Research Program supports this research through the National Research Foundation of Korea (NRF), funded by the Ministry of Education (NRF- RS-2023-00237287, NRF-2021S1A5A8062526) and local government-university cooperation-based regional innovation projects (2021RIS-003).

References

[1] Anupam Khan, and Soumya K. Ghosh, "Student Performance Analysis and Prediction in Classroom Learning: A Review of Educational Data Mining Studies," *Education and Information Technologies*, vol. 26, no. 1, pp. 205-240, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[2] Fadi Thabtah et al., "Data Imbalance in Classification: Experimental Evaluation," *Information Sciences*, vol. 513, pp. 429-441, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[3] Jafar Tanha et al., "Boosting Methods for Multi-Class Imbalanced Data Classification: An Experimental Review," *Journal of Big Data*, vol. 7, pp. 1-47, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[4] Mustafa Yağcı, "Educational Data Mining: Prediction of Students' Academic Performance Using Machine Learning Algorithms," *Smart Learning Environments*, vol. 9, no. 1, pp. 1-19, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[5] Ramin Ghorbani, and Rouzbeh Ghousi, "Comparing Different Resampling Methods in Predicting Students' Performance Using Machine Learning Techniques," *IEEE Access*, vol. 8, pp. 67899-67911, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[6] Sobhan Sarkar et al., "Predicting and Analyzing Injury Severity: A Machine Learning-Based Approach Using Class-Imbalanced Proactive and Reactive Data," *Safety Science*, vol. 125, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[7] G. Sambasivam, and Geoffrey Duncan Opiyo, "A Predictive Machine Learning Application in Agriculture: Cassava Disease Detection and Classification with Imbalanced Dataset Using Convolutional Neural Networks," *Egyptian Informatics Journal*, vol. 22, no. 1, pp. 27-34, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[8] Roweida Mohammed, Jumanah Rawashdeh, and Malak Abdullah, "Machine Learning with Oversampling and Undersampling Techniques: Overview Study and Experimental Results," *In Proceedings of the 2020 11th International Conference on Information and Communication Systems (ICICS)*, Irbid, Jordan, pp. 243-248, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[9] Keerthana Rajendran, Manoj Jayabalan, and Vinesh Thiruchelvam, "Predicting Breast Cancer Via Supervised Machine Learning Methods on Class Imbalanced Data," *International Journal of Advanced Computer Science and Applications*, vol. 11, no. 8, pp. 1-10, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[10] Fadel M. Megahed et al., "The Class Imbalance Problem," *Nature Methods*, vol. 18, no. 11, 1270-1272, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[11] Matloob Khushi et al., "A Comparative Performance Analysis of Data Resampling Methods on Imbalance Medical Data," *IEEE Access*, vol. 9, pp. 109960-109975, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[12] Ahmad S. Tarawneh et al., "Stop Oversampling for Class Imbalance Learning: A Review," *IEEE Access*, vol. 10, pp. 47643-47660, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[13] Shweta Sharma, Anjana Gosain, and Shreya Jain, "A Review of the Oversampling Techniques in Class Imbalance Problem," *Proceedings of the International Conference on Innovative Computing and Communications (ICICC)*, Springer Singapore, pp. 459-472, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[14] Jiawen Kong et al., "On the Performance of Oversampling Techniques for Class Imbalance Problems," *In Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining*, Springer, Cham, pp. 84-96, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[15] Fredy Rodríguez-Torres, José F. Martínez-Trinidad, and Jesús A. Carrasco-Ochoa, "An Oversampling Method for Class Imbalance Problems on Large Datasets, *Applied Sciences*, vol. 12, no. 7, pp. 1-17, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[16] Bin Liu, and Grigorios Tsoumakas, "Dealing with Class Imbalance in Classifier Chains via Random Undersampling," *Knowledge-Based Systems*, vol. 192, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[17] Debashree Devi, Saroj K. Biswas, and Biswajit Purkayastha, "A Review on Solution to Class Imbalance Problem: Undersampling Approaches," *Proceedings 2020 International Conference on Computational Performance Evaluation (ComPE)*, pp. 626-631, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[18] Shuhua Monica Liu, Jiun-Hung Chen, and Zhiheng Liu, "An Empirical Study of Dynamic Selection and Random Under-Sampling for the Class Imbalance Problem," *Expert Systems with Applications*, vol. 221, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[19] Khan Md. Hasib et al., "Imbalanced Data Classification Using Hybrid Under-Sampling with Cost-Sensitive Learning Method," *Proceedings of the Edge Analytics: Select Proceedings of 26th International Conference - ADCOM* 2020, Springer Singapore, pp. 423-435, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[20] Zhe Wang, Chenjie Cao, and Yujin Zhu, "Entropy and Confidence-Based Undersampling Boosting Random Forests for Imbalanced Problems," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 12, pp. 5178-5191, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[21] Muhammad Adil et al., "LSTM and Bat-Based RUSBoost Approach for Electricity Theft Detection," *Applied Sciences*, vol. 10, no. 12, pp. 1-21, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[22] Dhritiman Adhya, Soumesh Chatterjee, and Ajoy Kumar Chakraborty, "Diagnosis of PV Array Faults Using RUSBoost," *Journal of Control, Automation and Electrical Systems*, vol. 34, no. 1, pp. 157-165, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[23] Bhagat Singh Raghuwanshi, and Sanyam Shukla, "Classifying Imbalanced Data Using BalanceCascade-based Kernelized Extreme Learning Machine," *Pattern Analysis and Applications*, vol. 23, no. 3, pp. 1157-1182, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[24] Hao Chen et al., "Deep Balanced Cascade Forest: A Novel Fault Diagnosis Method for Data Imbalance," *ISA Transactions*, vol. 126, pp. 428-439, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[25] S.K. Gupta et al., "Data Imbalance in Landslide Susceptibility Zonation: Under-Sampling for Class-Imbalance Learning," *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 42, pp. 51-57, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[26] Nathalie Japkowicz, "Learning from Imbalanced Data Sets: A Comparison of Various Strategies," *Association for the Advancement of Artificial Intelignece Workshop Paper*, pp. 10-15, 2000. [Google Scholar] [Publisher Link]

[27] Gary M. Weiss, "Mining with Rarity: A Unifying Framework," *ACM SIGKDD Explorations Newsletter*, vol. 6, no. 1, pp. 7-19, 2004. [CrossRef] [Google Scholar] [Publisher Link]

[28] Jason Van Hulse, Taghi M. Khoshgoftaar, and Amri Napolitano, "Experimental Perspectives on Learning from Imbalanced Data," *ICML '07: Proceedings of the 24th International Conference on Machine Learning*, pp. 935-942, 2007. [CrossRef] [Google Scholar] [Publisher Link]

[29] Ricardo Barandela et al., "The Imbalanced Training Sample Problem: Under or Over Sampling?," *Structural, Syntactic, and Statistical Pattern Recognition*, pp. 806-814, 2004. [CrossRef] [Google Scholar] [Publisher Link]

[30] Hui Han, Wen-Yuan Wang, and Bing-Huan Mao, "Borderline-SMOTE: A New Over-Sampling Method in Imbalanced Data Sets Learning, Proceedings of the ICIC, 3644," *Advances in Intelligent Computing*, pp. 878-887, 2005. [CrossRef] [Google Scholar] [Publisher Link]

[31] J. Ross Quinlan, *C4.5: Programs for Machine Learning*, Morgan Kaufmann Publishers, San Francisco, CA, United States, 1993. [Google Scholar] [Publisher Link]

[32] William W. Cohen, "Fast Effective Rule Induction," *In Proceedings of the 12th International Conference on Machine Learning*, Tahoe City, California, pp. 115-123, 1995. [CrossRef] [Google Scholar] [Publisher Link]

[33] Johannes Fürnkranz, and Gerhard Widmer, "Incremental Reduced Error Pruning," *Proceedings of the International Conference on Machine Learning*, New Brunswick, NJ, pp. 70-77, 1994. [CrossRef] [Google Scholar] [Publisher Link]

[34] Tom M. Mitchell, *Machine Learning*, McGraw-Hill, pp. 1-28, 1997. [Google Scholar] [Publisher Link]

[35] Foster Provost, and Tom Fawcett, "Robust Classification For Imprecise Environments," *Machine Learning*, vol. 42, no. 3, pp. 203-231, 2001. [CrossRef] [Google Scholar] [Publisher Link]

[36] A.P. Bradley et al., "Precision–Recall Operating Characteristic (P-ROC) Curves In Imprecise Environments," *Proceedings of the 18th International Conference on Pattern Recognition*, Hong Kong, China, vol. 4, pp. 123-127, 2006. [CrossRef] [Google Scholar] [Publisher Link]

[37] Jesse Davis, and Mark Goadrich, "The Relationship Between Precision–Recall and ROC Curves," *Proceedings of the 23rd International Conference on Machine Learning*, pp. 233-240, 2006. [CrossRef] [Google Scholar] [Publisher Link]

[38] Yanmin Sun et al., "Cost-Sensitive Boosting for Classification of Imbalanced Data," *Pattern Recognition*, vol. 40, no. 12, pp. 3358-3378, 2007. [CrossRef] [Google Scholar] [Publisher Link]

[39] N.V. Chawla et al., "SMOTE: Synthetic Minority Oversampling Technique," *Journal of Artificial Intelligence Research*, vol. 16, pp. 321-357, 2002. [CrossRef] [Google Scholar] [Publisher Link]

[40] Yoav Freund, and Robert E. Schapire, "Experiments with A New Boosting Algorithm," *Proceedings of the 13th International Conference on Machine Learning*, pp. 148-156, 1996. [CrossRef] [Google Scholar] [Publisher Link]