Optimal Selection of Resampling Methods for Imbalanced Data with High Complexity

Annie Kim

The Graduate School
Yonsei University
Department of Biostatistics and Computing

Optimal Selection of Resampling Methods for Imbalanced Data with High Complexity

A Master's Thesis
Submitted to the Department of Biostatistics and Computing
and the Graduate School of Yonsei University
in partial fulfillment of the
requirements for the degree of
Master of Science

Annie Kim

June 2021

This certifies that the master's thesis of ***Annie Kim*** is approved.

_____

Inkyung Jung: Thesis Supervisor

_____

Chung Mo Nam: Thesis Committee Member #1

_____

Hyon Hee Kim: Thesis Committee Member #2

The Graduate School

Yonsei University

June 2021

Contents

List of Tables

List of Figures

Abstract

Optimal Selection of Resampling Methods

for Imbalanced Data with High Complexity

Class imbalance is considered to be a major problem in classification tasks. In the case of a class imbalance, the decision boundary is easily biased toward the majority class. The possible solutions for this can be divided into two groups: a data level solution (resampling) and an algorithm level solution. The resampling method is more adaptable in comparison with other methods since it can be used with various classifiers. However, some results have shown that oversampling, the most widely used resampling method, worsens classification performance. This is due to an overgeneralization problem. The overgeneralization problem occurs when examples produced by the oversampling technique are introduced into the majority class domain when they should be represented in the minority class domain. This paper claims that this overgeneralization problem is aggravated in complex data settings. To mitigate the problem of overgeneralization in complex datasets, two alternative approaches are provided. The first approach is to

incorporate the filtering method into oversampling. The second approach is to simply apply undersampling. In this study, the researchers investigated the relationship between complexity and imbalance for classification. Simulation studies and real data analysis were performed to compare resampling results in various scenarios that took into account different complexities, imbalances, and sample sizes. In conclusion, this study aids researchers in choosing the optimal resampling method for complex datasets.

---

key words: imbalanced data, overgeneralization, resampling, oversampling, undersampling

1. Introduction

The class imbalance problem is defined as the classification of datasets with unequal class distributions. Class imbalance problems occur frequently in many real-world tasks, such as fraud detection, spam detection, and cancer prediction. In these operations, non-events often occur, resulting in ‘not spam’, ‘not cancer’, and ‘not fraud’. Conversely, fraud, spam, and cancer do rarely occur. Even if these things happen less frequently, it does not mean that they are less important. Minority classes are generally more interesting in terms of learning tasks. Therefore, it is important to properly classify these infrequently occurring events.

However, imbalanced datasets constitute a challenge in many machine learning tasks. This is because most machine learning algorithms assume a roughly balanced class distribution. In the case of a class imbalance, the decision boundary is easily biased toward the majority class. Therefore, classification performance degrades when the imbalance ratio of the majority and minority classes is large. Several methods have been advised to resolve this issue. The solutions can be divided into two groups: data level solutions and algorithm level solutions. The data level solution, also known as the resampling method, changes the balance between the classes by modifying the data distribution. The solution to the algorithm level imposes a bias on the minority class by changing the algorithm's search technique.

The biggest advantage of the resampling method is that it is more versatile. One may preprocess the dataset once in order to use it to train different classifiers. The computation is only required to be done once to prepare the data in this way. Therefore, the resampling method can be used with various classification algorithms. As this is an advantage, numerous researchers are giving attention to this method and various methods of resampling have been developed along these lines.

The resampling method can be classified into undersampling and oversampling. Undersampling deletes instances from the majority classes to keep a balance between classes. This technique is known to provide a compact and balanced training set that reduces the cost of the learning process. However, important information may be lost. It also increases the variance of the classifier. On the other hand, oversampling balances the number of samples between classes by adding instance copies to the minority class or generating synthetic data. The most widely used method for oversampling is Synthetic Minority Oversampling Technique (SMOTE). It creates new, artificial data representing the minority classes in its own way.

However, even if SMOTE achieves a better distribution between classes, it can yield worse results than before. This is due to the overgeneralization problem. The overgeneralization problem is a situation in which examples produced by the oversampling techniques are introduced into the majority class domain when they should be represented in the minority class domain. This happens because SMOTE blindly oversamples without

taking into account other aspects that need to be considered. Due to this issue, non-minority class areas are incorrectly classified as minority classes.

This overgeneralization problem can be aggravated in complex data settings. According to the work done by Ho and Basu (2002), three factors can be attributed to define complex data: the ambiguity of the class, the sparsity of the data and the complexity of the boundaries separating the classes. Using oversampling on complex data settings can result in creating unnecessary minority class examples that do not ease the learning of the minority class. It can also make the boundaries between classes unclear, making the classifier unusable. For this reason, it can be understood how data complexity is a related issue affecting overgeneralization.

To mitigate the problem of overgeneralization in complex datasets, this paper advises the use of two alternative approaches. The first approach is to incorporate the filtering method into oversampling. Adding a filter to the oversampling method can reduce the problems created by noise and borderline examples in the complex datasets. SMOTE is often used in conjunction with filtering methods to remove examples that are considered harmful for classification. SMOTE-Tomek Links and SMOTE-ENN are two classic techniques that add filters to the original SMOTE.

Another solution is to simply apply an undersampling method. Applying the undersampling method avoids adding complexity to an existing complex datasets, and one can also filter out noisy or borderline examples from the existing datasets. In addition,

according to work done by Park and Jung (2009), the undersampling method performed better than other resampling methods in spite of a loss of information due to undersampling.

The main purpose of this paper is to provide the optimal resampling methods for imbalance in complex datasets. For this purpose, the researchers investigated the relationship between data complexity and resampling method choices. To achieve this, six oversampling, ten undersampling and ten filtering methods were applied to various simulations and real data, considering the range of sample size, imbalance ratio and data complexity.

This paper is organized as follows. In section 2, brief definitions of the complex datasets are explained. In section 3, the resampling methods are introduced. In sections 4 and 5, the resampling methods are applied to the aforementioned simulated and real life datasets, and the results of the classification are analyzed. Through this, the optimal resampling methods according to the complexity of the dataset are suggested. Finally, in section 6, some conclusions are drawn.

2. Definition of Complex Data

The empirically observed resampling method is highly dependent on data characteristics. It is believed that the study of the relationship between data characteristics and resampling methods is essential to improving the classification performance of imbalanced datasets and further developing the resampling methods. In this section, the data characteristics are defined, and using these characteristics, the optimal resampling methods are provided in a later section.

Most classification problems suffer from various degrees of difficulty. From the work done by Ho and Basu (2002), this difficulty is characterized using three components: (1) class ambiguity, (2) data sparsity, and (3) complexity of the boundaries separating the classes.

Class ambiguity, the first factor of data complexity, comes from scenarios where classes cannot be distinguished using given data, regardless of the classification algorithm used. This is the problem when involving non-discriminatory features. An example of a non-discriminatory feature is described in the first plot of Figure 1. As the overlap of the distribution of the feature values within a class increases, the classification task is considered to be more difficult.

Data sparsity, the second factor of data complexity, is when certain expected values in the dataset are missing. The sparsity of the data can be related to the imbalance problem. After training, subsequent areas are randomly classified. As a result, the ability to perform accurate prediction alters. Examples of sparse dataset can be seen in the second plot of Figure 1. The test data in the grey area will be classified randomly after training.

The complexity of the decision boundary, the third factor of data complexity, is influenced by the borderline and noisy examples. The borderline examples are samples in the area near the class boundaries. The presence of the borderline example contributes to the accuracy of the decision boundary. The noisy example is an individual of one class which occurs in the area of another class. Zhu and Wu (2004) describe the cause of noisy data as the subjectivity of labeling or an inappropriate labeling process. As shown in the third plot of Figure 1, the classes overlap and examples are very close to the decision boundary.

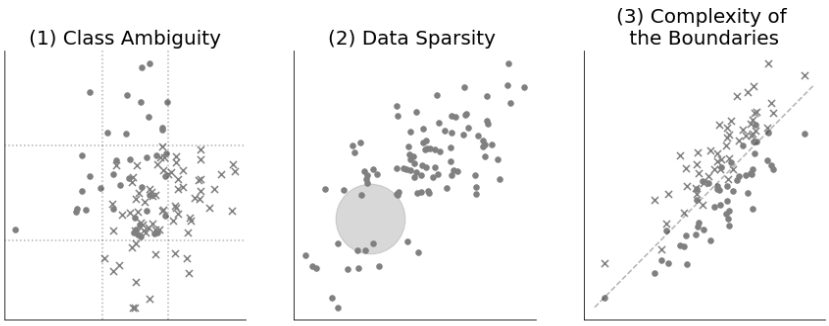


Figure 1. Three Factors Attributing to Data Complexity

3. Resampling Methods

In this section, classic oversampling and undersampling methods are introduced, which are widely used. Then, a recently proposed filtering-based oversampling method is introduced as one of the methods to solve the overgeneralization problem. All resampling methods used are summarized in Table 1.

Table 1. Resampling Methods

Oversampling	Undersampling	Filtering
Random Oversampling	Random Undersampling	SMOTE-TL
SMOTE	Near Miss	SMOTE-ENN
ADASYN	Tomek Link	DSRBF
Borderline SMOTE	CNN	TRIM-SMOTE
SVM SMOTE	ENN	SMOTE-RSB*
Kmeans SMOTE	RENN	NRSBoundary
	ALL KNN	NEATER
	OSS	SMOTE-IPF
	NCR	SMOTE-FRST-2T
	IHT	NRAS

3.1 Oversampling Methods

Oversampling balances the number of examples between classes by adding an instance copy of an underrepresented class or generating artificial data. The researchers introduced three oversampling methods. In addition, three modified versions of the existing oversampling methods have been introduced.

3.1.1 Classic Oversampling Methods

• **Random Oversampling**

Random Oversampling is the most naive strategy for generating new examples. It randomly samples current available examples with replacement. As a result, the number of examples from the majority and minority classes is balanced, and the majority class does not take over another class during training. However, repeated sampling can lead to overfitting problems. If repeating samples are an issue, the algorithm allows the users to control the dispersion of randomly selected examples. This paper applied an original method of random oversampling.

• **Synthetic Minority Oversampling Technique (SMOTE)**

In addition to the random oversampling, there are two very popular ways to oversample a class of minority. Of those two methods, SMOTE can be illustrated with Figure 2. First, it randomly selects a minority class instance as the basis for generating new synthetic data.

Then, the closest neighbors of the same class are selected. Finally, random interpolation is performed between two points to obtain a new minority class instance. In Figure 2, SMOTE selects instances x_{zi} that are close to the randomly selected minority class example x_i. New artificial example x_{new} is generated by random interpolation between two instances.

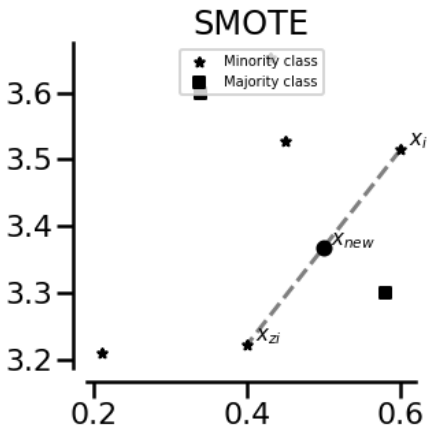


Figure 2. Visual Description of SMOTE

• **Adaptive Synthetic Sampling Technique (ADASYN)**

ADASYN automatically determines the number of examples to be oversampled for each minority class data by considering distribution of dataset. The example that needs to be oversampled is determined by the difficulty of learning. The difficulty of learning can be quantified through the ratio of values belonging to the majority class among the k-nearest neighbors of values belonging to the minority class.

3.1.2 Oversampling Variants

The oversampling methods introduced the problem of overgeneralization. It blindly generates the minority class example without taking into account the majority class. This is especially problematic for highly complex datasets. In such cases, applying the oversampling could generate unnecessary synthetic data that do not facilitate the learning process. In addition, synthetic data could be dedicated to unclear decision boundaries between classes. Various methods have been developed to solve this problem.

- **Borderline SMOTE**

Borderline SMOTE determines the best candidates to oversample in the overall dataset before oversampling. This algorithm oversamples with examples close to the decision boundary, which comes from the premise that examples far from the boundary may have little contribution to classification success. In order to identify the decision boundary, this algorithm uses the ratio between the majority and minority class within the neighborhood of each minority class to be oversampled. To not consider noise samples in the oversampling process, examples with all neighbors of the majority class are not oversampled.

- **SVM SMOTE**

SVM SMOTE is another variant of Borderline SMOTE. While Borderline SMOTE uses the k-nearest neighbor to detect decision boundaries, SVM SMOTE uses support vector

machine algorithms. The boundary area is approximated with a support vector after training with the SVM. Synthetic examples are randomly generated along the line connecting each minority class support vectors with its nearest neighbor. More or fewer support vectors can be selected by controlling the SVM classifier's C parameter.

• **Kmeans SMOTE**

Kmeans SMOTE uses the k-means clustering before applying SMOTE. K-means clustering groups examples together and creates new examples based on clustering results. Kmeans SMOTE not only prevents noise from oversampling, but also pays attention to imbalances between and within clusters.

3.2 Undersampling Methods

Oversampling blindly creates a class of minority examples without taking into account other aspects. In return, an overgeneralization problem occurs. In case of the overgeneralization problem, the researchers suggested applying an undersampling method instead. Undersampling is an efficient technique that does not add new data. Undersampling reduces the risk of creating false decision boundaries created by artificial examples.

All the undersampling methods introduced in this section selects samples from the original dataset, S. So the result of undersampling, S', is $S' \subset S$. There are two types of undersampling. The controlled undersampling technique allows a user-specified

undersampling strategy to specify the number of samples in S'. The cleaning undersampling technique does not allow this specification. Two controlled undersampling methods and eight cleaning undersampling methods are introduced in this section.

3.2.1 Controlled Undersampling Techniques

• Random Undersampling

Random Undersampling involves selecting an example randomly from majority class with or without replacement. This is the simplest strategy for an imbalanced classification problem as with random oversampling. However, if imbalance is severe, that level of information loss may occur. Therefore, performance may vary depending on the degree of imbalance. In this paper, the majority class examples will be undersampled until it matches the number of examples in the minority class.

• Near Miss

Unlike Random Undersampling, which randomly selects majority class samples without any rules, Near Miss adds some heuristic rules. There are three versions of the Near Miss algorithm which are explained in Figure 3. NearMiss-1 selects the majority class example with the smallest mean distance from the minority class to the nearest N samples. NearMiss-2 selects the majority class sample with the smallest average distance from the minority class to the furthest N samples. NearMiss-3 is a two-stage algorithm that mixes NearMiss-1 and NearMiss-2. First, for each majority class example, M nearest neighbors

are kept. Finally, the majority class examples selected are those with the largest average distance to the N nearest neighbors. This paper used NearMiss-3 because of its robustness to noise over other versions of NearMiss.

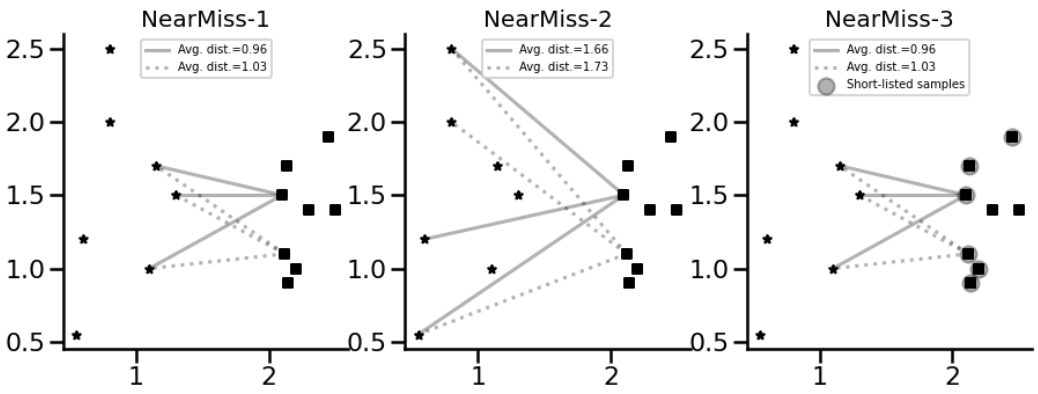


Figure 3. Three Versions of Near Miss

3.2.2 Cleaning Undersampling Techniques

Cleaning undersampling techniques does not allow the user to set the number of samples to undersample. Therefore, after resampling, the number of minority and majority class samples is different.

- **Tomek Link**

Tomek Link method does undersampling by detecting Tomek's link in majority class examples and removing it. The definition of Tomek's link is described in Figure 4: Let

$\mathrm{d}(.)$ be the distance between two examples. If x and y are from different classes and there is no sample z that satisfies the following condition $\mathrm{d}(\mathrm{x},\mathrm{y}) < \mathrm{d}(\mathrm{x},\mathrm{z})$ or $d(x,y) < d(y,z)$, then the pair of x, y is a tomek link. Values classified as Tomek's link are considered noise or borderline examples.

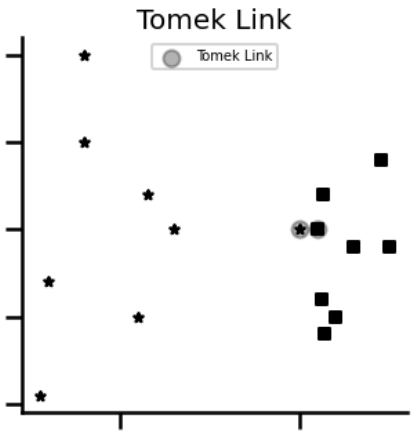


Figure 4. Visual Description of Tomek Link

• Condensed Nearest Neighbors (CNN)

CNN iteratively determines whether or not the example should be removed using the 1-nearest neighbor rule. The purpose of CNN is to find the smallest subset representing the majority class. However, CNN are known to be noise sensitive, and as a result add noisy samples to the resampling result.

• Edited Nearest Neighbors (ENN)

ENN edits the dataset by removing examples that do not match with its neighbors. This

algorithm is an extension of CNN that uses 3-nearest neighbors instead of 1. For each minority class example, the 3-nearest neighbors are computed and if the selection criterion is not satisfied, the example is undersampled. Few variations of ENN have been introduced in the following. These ENN variants added few rules to the original ENN.

• **Repeated Edited Nearest Neighbors (RENN)**

RENN extended ENN by repeating the algorithm multiple times. RENN repeats ENN until the dataset to be undersampled converges. Generally, repeating the algorithm will delete more examples compared with ENN.

• **ALL KNN**

ALL KNN extended RENN by increasing the number of nearest neighbors at each iteration. It increases K to 1, 3, and 5 and includes those examples that are satisfied in all iterations. It is known to be conservative compared to other methods. But ENN, RENN and ALL KNN have similar impact on imbalanced datasets since it cleans noisy examples next to the decision boundaries.

• **One Sided Selection (OSS)**

OSS is an extension of ENN. It uses Tomek's links and 1-nearest neighbor rule to remove noisy samples. It first eliminates tomek links then applies 1-nearest neighbor rule to undersample misclassified examples.

• **Neighborhood Cleaning Rule (NCR)**

NCR is an extension of ENN. It focuses on cleaning the dataset rather than condensing them. When the value is classified as a majority class using 3-nearest neighbors, the value is deleted. When the value is classified as a minority class, majority class examples in 3-nearest neighbors are deleted.

• **Instance Hardness Threshold (IHT)**

IHT uses a trained classifier algorithm to remove the examples with low predictive probabilities. Unlike other methods that use a distance based classification method, this algorithm applies predictive probability based on classification model. Using a classification model, the probability that the model is difficult to classify is calculated. The training of the classifier is performed using a cross-validation. For this paper, a Random Forest classifier with 5-fold cross validation was used.

3.3 Filtering Methods

Another solution suggested, when dealing with the overgeneralization problem, was adding filters to oversampling methods. By cleaning the space resulting from oversampling, the overgeneralization problem can be resolved. In this section, ten filtering methods are introduced.

• **SMOTE-Tomek Link & SMOTE-ENN**

Tomek's link and edited nearest neighbors are both cleaning methods that have been added to the pipeline after applying SMOTE to obtain a cleaner space. SMOTE-Tomek Link and SMOTE-ENN are the most typical techniques of filtering methods. These methods are motivated by SMOTE's well known drawback of generating noisy examples.

Filtering of artificially generated examples is a frequent operation that supports the success of SMOTE on real data. The variants of filtering methods have been developed to enhance classification performance in imbalanced datasets. All the following methods attempt to solve the overgeneralization problem by applying new innovative methods to the existing oversampling.

• **Dynamic SMOTE Radial Basis Function (DSRBF)**

This procedure incorporated SMOTE with a memetic algorithm (MA) that optimizes radial basis functions neural networks (RBFNNs), a clustering process, and local search procedure. DSRBF runs the oversampling procedure in the preprocessing stage and in different MA generations.

• **TRIM-SMOTE**

TRIM is utilized as a preprocessing method for oversampling methods that are developed based on SMOTE. TRIM is divided into two stages: minority class clustering stage and cluster connection stage.

In the minority clustering stage TRIM recursively splits the entire dataset into smaller clusters. Although both majority and minority class data can be contained in a cluster, most of the majority samples are pruned. In this stage, although each minority class cluster is well-approximated, some clusters can be considered as noise. User-defined threshold on minimum precision is used to prune the noise clusters.

• **SMOTE-RSB***

This method introduced a new undersampling method over the generated artificial examples for highly imbalanced datasets. It first builds new synthetic examples of minority class using SMOTE. Then, in order to improve the quality, the selection method based on the rough set theory is used. The process contains removing generated examples that do not associate with the lower approximation of the minority class, since examples in the boundary region are noisy data that do not benefit the improvement of classification performance.

• **NRSBoundary**

NRSBoundary is an extension of SMOTE-RBS*. It divides the dataset into three groups. And out of these three groups, the boundary region indicates the overlap between the majority and minority class examples. It only oversamples the minority class examples in the boundary region.

• **NEATER**

NEATER is a non-cooperative game where the goal of the game is to label all artificially generated examples. This approach does not consider generated artificial data as the minority class examples. Rather, it keeps generated examples unlabeled. It determines most-likely class though non-cooperative game strategy. All the generated artificial data that do not belong to the minority class are removed.

• **SMOTE-IPF**

SMOTE-IPF applies an IPF filter to remove the noisy examples in the original dataset and those created by SMOTE. IPF removes noisy examples in multiple iterations. The stopping criteria is when, for a number of iterations, the number of recognized noisy examples in each of the iterations is less than a specific percentage.

• **SMOTE-FRST-2T**

SMOTE-FRST-2T modified SMOTE-FRST algorithm in order to improve solving imbalance problems. SMOTE-FRST uses fuzzy rough set theory in order to remove data that do not belong to the majority class region. However, setting a suitable threshold for this method is difficult since too high threshold eliminates too many examples and too low threshold undermines the cleaning process. SMOTE-FRST-2T uses a double threshold for eliminating original majority class and synthetic examples.

- **Noise Reduction A Priori Synthetic Oversampling Technique (NARS)**

NRAS improved prediction of under-represented samples containing noise by removing outliers resulting in statistically significant improvement. NARS uses Bayes Theorem to calculate the probability of group membership. It includes minority class examples that do not appear as noise.

4. Evaluation of Resampling Methods in the Simulated Complex Dataset

To investigate the relationship between the data characteristics and selection of resampling methods, simulated data was created with various combinations of concepts of complexity, training set sizes, and degrees of imbalance. Different kinds of resampling were applied to the generated data. The results showed the relationship between the data characteristics and selection of resampling methods, which as a result allowed for the selection of optimal resampling methods for each data characteristic.

4.1 Simulation Framework

The generation method used was inspired by Japkowicz and Stephen (2002) who designed a similar framework for testing the effect of resampling in complex settings. However, their works were only generated five times for each simulated domain, so the results were not reliable. Also the performance measures used were of a single threshold value, which may vary by the choice of the threshold. Therefore, this paper aimed to provide more precise results with better performance metrics. The simulated data of this study were generated in the following way:

$$\begin{bmatrix} X_1 \\ X_2 \\ X_3 \\ \widetilde{x_4} \\ \widetilde{x_5} \\ \widetilde{x_6} \end{bmatrix} \sim MVN\left(\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 & \rho & \rho & \rho & \rho & \rho \\ \rho & 1 & \rho & \rho & \rho & \rho \\ \rho & \rho & 1 & \rho & \rho & \rho \\ \rho & \rho & \rho & 1 & \rho & \rho \\ \rho & \rho & \rho & \rho & 1 & \rho \\ \rho & \rho & \rho & \rho & \rho & 1 \end{bmatrix} \right), \quad where \ \rho = 0.3$$

$$\mathrm{X}_4 = \begin{cases} 0 & if \ \widetilde{x_4} < \Phi^{-1}(0.3) \\ 1 & o.w \end{cases}$$

$$\mathrm{X}_5 = \begin{cases} 0 & if \ \widetilde{x_5} < \Phi^{-1}(0.2) \\ 1 & o.w \end{cases}$$

$$\mathrm{X}_6 = \begin{cases} 0 & if \ \widetilde{x_6} < \Phi^{-1}(0.15) \\ 1 & o.w \end{cases}$$

$$\eta = {}^{1}/_{1 + \exp(-1 * (1.1\mathrm{X}_1 + 0.9\mathrm{X}_2 + 0.7\mathrm{X}_3 + \mathrm{X}_4 + \mathrm{X}_5 - X_6 + \epsilon))} \quad \epsilon \sim N(0,1)$$

A total of six independent variables were generated. Three continuous variables and three categorical variables were created, and a correlation of 0.3 was established between the variables. Then, the sigmoid function was applied to the generated data in order to make η lay between 0 and 1. Japkowicz and Stephen's backbone model was applied to η to control the complexity, imbalance and sample size in the binary problems.

Three different complexity levels were used where level c corresponded to a backbone model composed of 2^{c} regular intervals. For example, the domain generated at the $\mathrm{c} = 1$ divided η by 0.5. Furthermore, when the $\mathrm{c} = 2$, η was divided as shown in Figure 5.

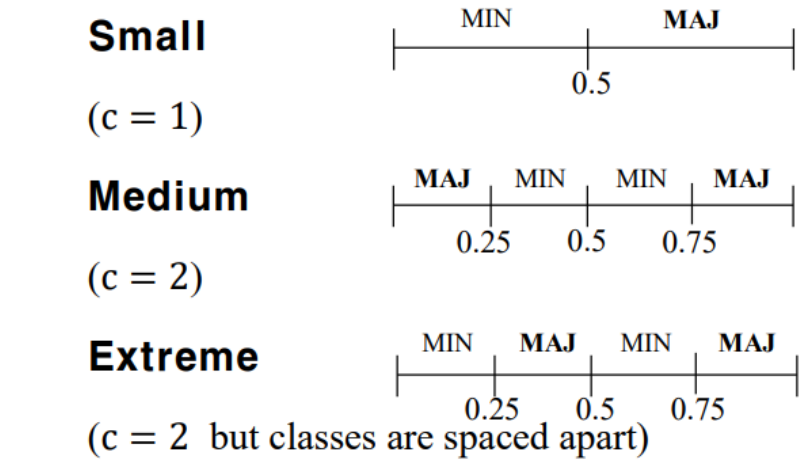


Figure 5. Three Complexity Levels

Three dataset sizes were considered. Each regular interval was, in fact, represented by $\text{round}((\left(\frac{5000}{32}\right) * 2^{s})/2^{c})$ training points. However, this was before the imbalance factor was considered.

Finally, three levels of class imbalance were considered where each level, i, corresponded to the situation where each subinterval of class 1 was represented by all the data it was normally entitled to (given c and s), but each subinterval of class 0 contains only $1/(32/2^{i})$th of all its normally entitled data. Each of the subintervals of class 0 were represented by $\text{round}\left(\frac{\left(\left(\frac{5000}{32}\right)*2^{s}\right)/2^{c}}{32/2^{i}}\right)$ training examples.

To summarize, with three levels of sample size, imbalance ratio, and complexity ($c = 1, 2, s = 1, 3,\ 5,\ i = 1, 3, 5$), sample size of each majority class interval is:

$$\left(\left(\frac{5000}{32}\right) * 2^{s}\right)/2^{c}$$

sample size of each majority class interval is:

$$\frac{\left(\left(\frac{5000}{32}\right) * 2^{s}\right)/2^{c}}{32/2^{i}}$$

By controlling for complexity (complexity=small, medium, extreme), imbalance (i=1, 3, 5) and size (s=1, 3, 5), 27 domains were generated. Each domain was generated 50 times. The details of the simulation framework are described in the table below. From Table 2, it can be seen that the degree of imbalance has IRs of 16, 4, and 1, respectively, and the sample size is approximately 180 in the smallest case, about 700 in the middle, and 2700 in the largest case. Although when IR is 1 resampling is not necessary, the following domains were set to show the difference in performance depending on the degree of imbalance.

Table 2. Description of Simulation Settings

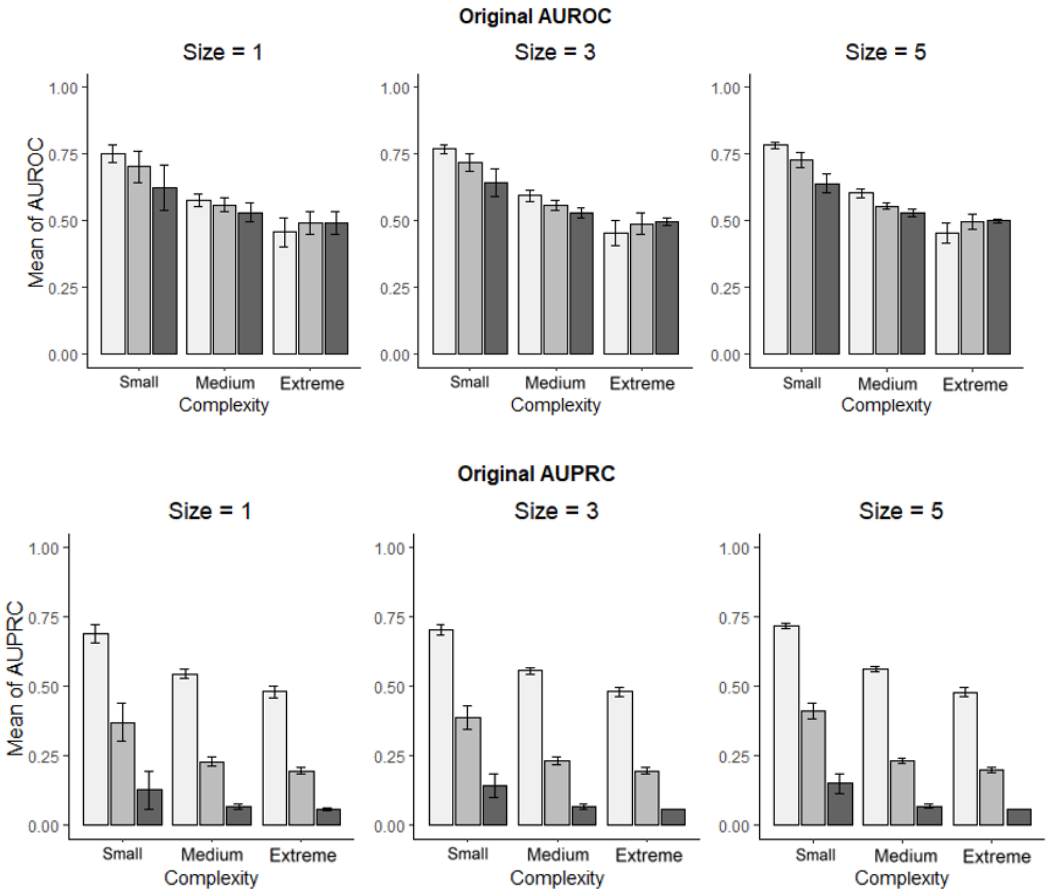
c	s	i	N	N+	IR
1, 2	1	1	166	156	15.6
		3	195	156	4
	3	1	664	625	16
		3	781	625	4
	5	1	2656	2500	16
		3	3125	2500	4

4.2 Other Setup Details

As most of the resampling results performed similarly when applied to various classification algorithms, this simulation only used the decision tree classifier as a classification method. The testing sets were created independently considering the complexity and imbalance ratio of each domain, and the size of the testing sets were set to 5000. The parameters for the decision tree classifier were chosen by 3-fold cross validation.

For evaluation, two performance metrics were used, namely the Area under the precision recall curve (AUPRC) and the area under the receiver operating characteristic (AUROC). In the prior research, selecting a suitable performance evaluation method has often been underestimated. Most of the works related to resampling methods compare methods using single-threshold measures such as accuracy, F-measure or G-score. This can be misleading since those measures can vary by the selection of the threshold. A powerful key is to use threshold free measures like AUROC or AUPRC.

The AUROC is the most popular measure for classification. The ROC uses a false positive rate for the x-axis and a true positive rate (sensitivity) for the y-axis. However, using this measure can present an overly optimistic view of a classification performance if there is a large class imbalance. Therefore, in imbalanced settings, considering both the AUROC and AUPRC helps to avoid misinterpretation. Figure 6 shows the AUROC and AUPRC of the decision tree classifier when the resampling method was not applied in each domain. Through this, it can be confirmed that the AUPRC rather than the AUROC showed a better performance difference according to the degree of imbalance.



Colors indicate imbalance; light gray bars are less imbalanced than dark bars

Figure 6. AUROC and AUPRC of Decision Tree Classifier in Each Domain

The PR curve uses recall (sensitivity) as the x-axis and precision as the y-axis. The goal in the ROC is to be in the upper left corner and in the PR curve the goal is to be in the upper right corner. The difference between the AUROC and AUPRC is true negative (TN) in the confusion matrix. Differences between the ROC curve and PR curve exist when in an imbalanced dataset because the number of negative examples greatly exceeds the number of positive examples. PR curves can uncover differences that are not visible in the ROC.

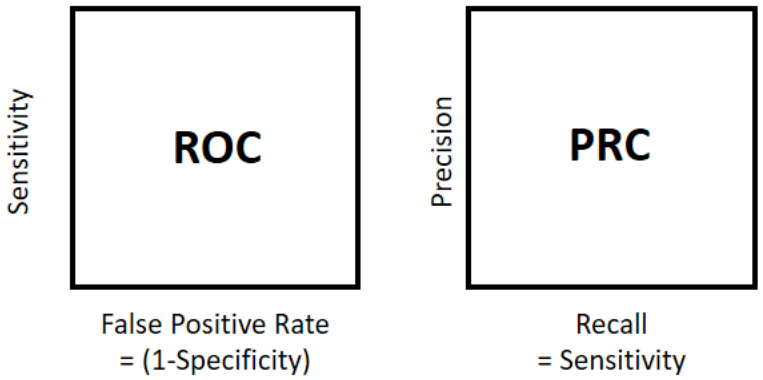
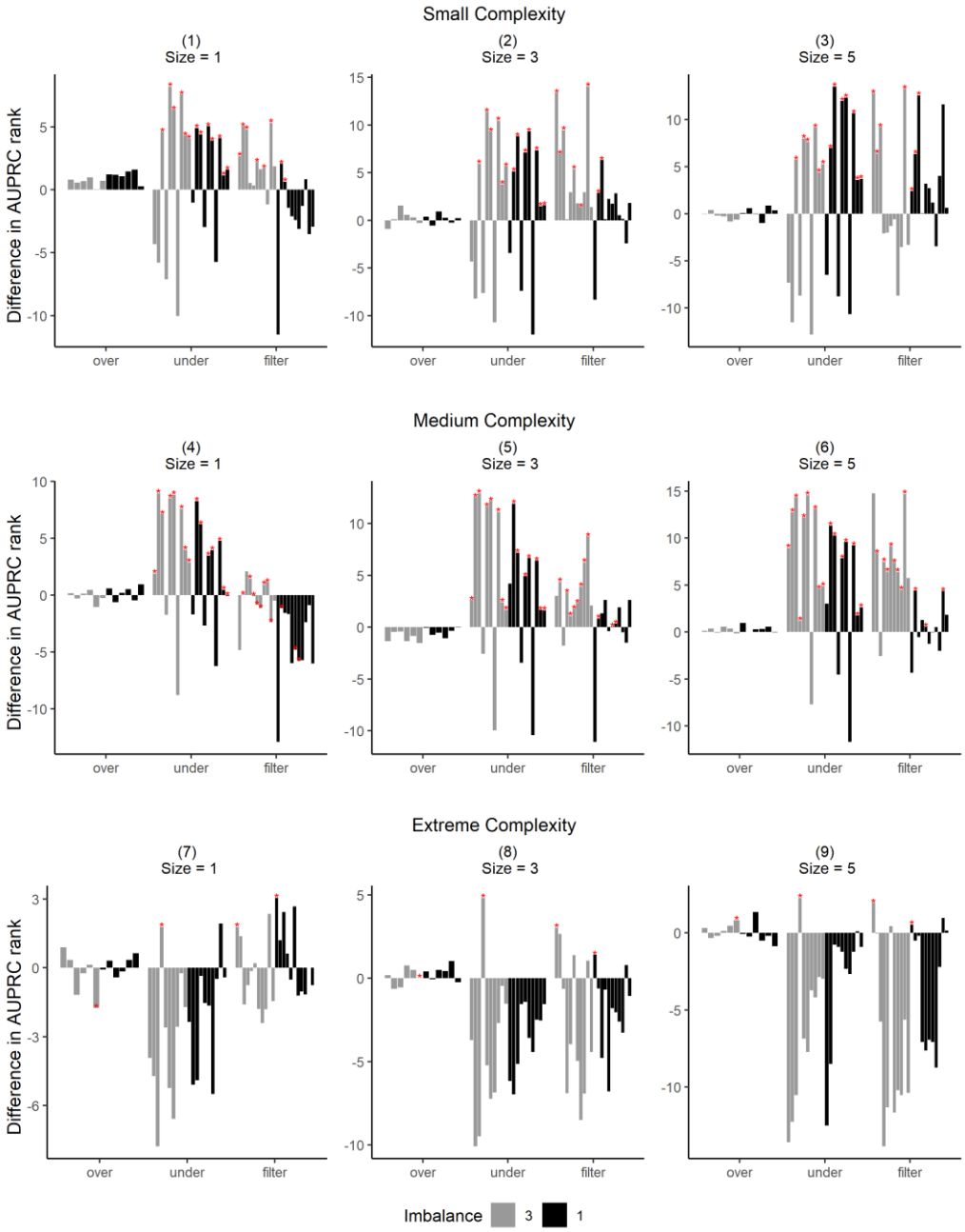


Figure 7. The Difference Between the ROC and PR Curve

4.3 Results

The simulation results are illustrated by the two figures below. Each figure describes the difference in ranking after resampling was applied. 0 indicates the ranking of the original classification result, and positive/negative value indicates increase/decrease of ranking after resampling. From these results, the researchers could demonstrate some behavior of the resampling technique on complex and imbalanced datasets.

Each plot in Figure 8-9 has a different data complexity and size. Furthermore, in each plot, the x-axis represents each resampling method described in this paper, and the y-axis represents the mean difference of the performance measures. Colors indicate imbalance; light gray bars are less imbalanced than dark bars. The star marker at the top of the bar indicates a significant performance gain through a paired t-test.



Star marker at the top of the bar indicates a significant performance gain.

Figure 8. The Difference in AUPRC Rank

4.3.1 Results of AUPRC

The first thing the researchers discovered through Figure 8 was oversampling showed a lower rank in all cases, regardless of the complexity and imbalance. Oversampling was expected to have a decrease in performance as complexity increases due to overgeneralization problems. However, simulated results showed that these problems occur even when the data are relatively less complex. Therefore, the researchers concluded that creating incorrect artificial data to balance between classes could contribute to performance decrement in all complexity ranges. However, compared to other resampling methods, oversampling showed more robust results without rapid rank changes even at extreme complexity. Also, the ranking differences were not visible according to the degree of imbalance.

Another result was that undersampling showed the highest rank among three resampling categories along with filtering methods. This allowed the researchers to infer that noisy or borderline examples from original data or oversampling greatly affect classification performance.

When the data complexity is small or medium, undersampling had the upper rank in most cases, and in the case of extreme data complexity, filtering methods had the upper rank. This can be inferred that, in case of extreme data complexity, the information loss due to undersampling had a large effect.

Furthermore, the results of undersampling showed a difference in rank depending on

the imbalance ratio. In case of the severe imbalance cases, the ranking was low compared to less severe cases. This is because there were more majority class examples to be removed when imbalance was severe. However, looking at plot (3), (6), and (9) of Figure 8, when the sample size was large, there were little difference in ranking due to imbalance.

Lastly, the filtering method had the highest ranking for extreme data complexity. However, as shown in plot (4) and (7) of Figure 8, when the sample size was small, the data was severely imbalanced, and there was small or medium complexity, the application of the filtering methods led to decrease in the ranking.

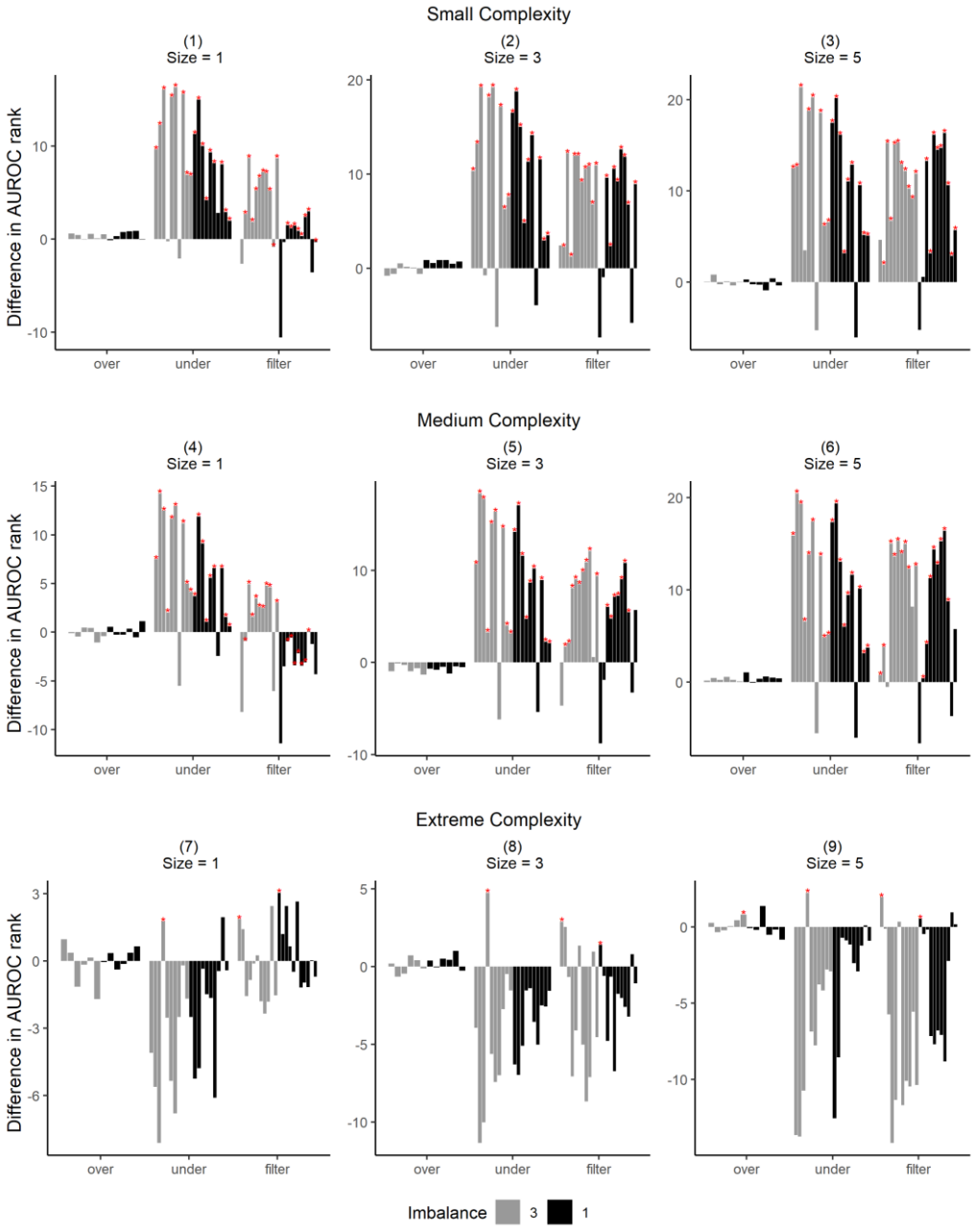
4.3.2 Comparing Results of AUROC and AUPRC

Figure 9, which is the result of AUROC, is almost similar to Figure 8, the result of AUPRC. However, the difference according to the selection of the performance matrix can be summarized in the following.

The first difference is that the ranking increased in the AUROC but decreased in the AUPRC. Looking at the plot (1) of Figures 8 and 9, it showed that the application of the filtering method significantly improved the AUROC, but the results of AUPRC mostly decreased.

The second difference is that the performance due to imbalance varied according to the selection of the performance matrix. When the AUPRC was used, the differences according to imbalances appeared, but looking at the AUROC result, no difference was shown. From plot (5) and (6) of Figure 8, it showed that the rankings were low when the imbalance was

severe, but from plot (5) and (6) of Figure 9, the rankings according to imbalance were similar.



Star marker at the top of the bar indicates a significant performance gain.

Figure 9. The Difference in AUROC Rank

The third difference is the selection of optimal resampling methods. In the plot (3) of Figure 8, when the imbalance was 3, the ranking of the filtering method seemed to be most optimal, but from plot (3) of Figure 9, which is the result of AUROC, the ranking of the undersampling method seemed higher than that of the filtering method.

Because of these differences, it is important to use an appropriate performance matrix according to the problem defined.

5. Evaluation of Resampling Methods in Real Data

In this section, the performance of six oversampling, ten undersampling and ten filtering methods using four classification algorithms were compared. In the process, *'complexity measures'* were used to represent data characteristics. Through this, the characteristics of the resampling method according to various data complexities were shown and the optimal combination of classifier and resampling was presented.

5.1 Data Description

109 labeled datasets from the University of California Irvine (UCI) machine learning repository were collected for the experiment. Most of these were based on multiclass datasets reorganized into binary problems by selecting some classes to compose the minority class and considering all other classes as the majority class. Some datasets were generated artificially, most of them are real life problems. Table 3 summarized the characteristics of these datasets: the number of samples, the number of features, the number of majority class examples and the imbalanced ratio. The table is sorted by the imbalance ratio (IR).

$$\text{IR} = \frac{\#\ of\ \ examples\ \ in\ \ minority\ \ class(+N)}{\#\ of\ \ examples\ \ in\ \ majority\ \ class(-N)}$$

Table 3. Summary of Real Data Characteristics

Name	N	d	+N	IR	Name	N	d	+N	IR	Name	N	d	+N	IR
Glass1	214	9	138	1.82	Ecoli 0-1 vs 2-3-5	244	7	220	9.17	Lymphography normal	148	23	142	23.67
Wisconsin	683	9	444	1.86	Ecoli 0-2-6-7 vs 3-5	224	7	202	9.18	Flare F	1066	11	1023	23.79
Pima	768	8	500	1.87	Glass 0-4 vs 5	92	9	83	9.22	Car good	1728	6	1659	24.04
Ecoli 0 vs 1	219	7	143	1.88	Ecoli 0-3-4-6 vs 5	205	7	185	9.25	Car vgood	1728	6	1663	25.58
Iris0	150	4	100	2	SATIMAGE	6435	36	5809	9.28	Kr vs K zero one vs draw	2901	6	2796	26.63
Glass0	214	9	144	2.06	Ecoli 0-3-4-7 vs 5-6	257	7	232	9.28	Kr vs K one vs fifteen	2244	6	2166	27.77
German	1000	29	700	2.33	Yeast 0-5-6-7-9 vs 4	528	10	477	9.35	Yeast4	1484	10	1433	28.1
Yeast1	1484	10	1055	2.46	Vowel 0	988	13	898	9.98	Winequality red 4	1599	11	1546	29.17
Haberman	305	3	225	2.81	Ecoli 0-6-7 vs 5	220	6	200	10	Poker 9 vs 7	244	25	236	29.5
Vehicle1	845	18	627	2.88	Glass 0-1-6 vs 2	192	9	175	10.29	Kddcup guess	1642	38	1589	29.98
Vehicle3	846	18	629	2.9	Ecoli 0-1-4-7 vs 2-3-5-6	336	7	307	10.59	Yeast 1-2-8-9 vs 7	947	10	917	30.57
Vehicle2	846	18	634	2.99	Led7digit 0-2-4-6-7-8-9	443	7	406	10.97	Abalone 3 vs 11	502	8	487	32.47
ADA	4147	47	3118	3.03	Ecoli 0-1 vs 5	240	6	220	11	Winequality white 9 vs 4	168	11	163	32.6
Glass 0-1-2-3 vs 4-5-6	214	9	163	3.2	Glass 0-6 vs 5	108	9	99	11	Yeast5	1484	10	1440	32.73
Vehicle 0	846	18	647	3.25	Glass 0-1-4-6 vs 2	205	9	188	11.06	Kr vs K three vs eleven	2935	6	2854	35.23

Ecoli1	335	7	258	3.35	Glass2	214	9	197	11.59	Winequality red 8 vs 6	656	11	638	35.44
Hepatitis	155	19	123	3.84	Ecoli 0-1-4-7 vs 5-6	332	6	307	12.28	Ecoli 0-1-3-7 vs 2-6	281	7	274	39.14
SPECT_F	267	44	212	3.85	Cleveland 0 vs 4	177	23	164	12.62	Abalone 17 vs 7-8-9-10	2338	8	2280	39.31
New thyroid1	215	5	180	5.14	Ecoli 0-1-4-6 vs 5	280	6	260	13	Abalone 21 vs 8	581	8	567	40.5
KC1	2109	21	1783	5.47	PC1	1109	21	1032	13.4	Yeast6	1484	10	1449	41.4
Ecoli2	335	7	284	5.57	Shuttle c0 vs c4	1829	9	1706	13.87	Winequality white 3 vs 7	900	11	880	44
Segment0	2308	23	1979	6.02	Yeast 1 vs 7	459	7	429	14.3	Winequality red 8 vs 6-7	855	11	837	46.5
Glass6	214	9	185	6.38	Sylva	13086	212	12281	15.26	Kddcup land vs portsweep	1061	40	1040	49.52
Yeast3	1484	10	1321	8.1	Glass4	214	9	201	15.46	Abalone 19 vs 10-11-12-13	1622	8	1590	49.69
Ecoli3	336	7	301	8.6	Ecoli4	335	7	315	15.75	Kr-vs-K zero vs eight	1460	6	1433	53.07
Page blocks0	5472	10	4913	8.79	Page blocks 1-3 vs 4	472	10	444	15.86	Winequality white	1482	11	1457	58.28
Ecoli 0-3-4 vs 5	200	7	180	9	Abalone9 18	731	8	689	16.4	Poker 8-9 vs 6	1485	25	1460	58.4
Yeast 2 vs 4	514	8	463	9.08	Dermatology6	358	129	338	16.9	Shuttle 2 vs 5	3316	9	3267	66.67
Ecoli 0-6-7 vs 3-5	222	7	200	9.09	Zoo3	101	16	96	19.2	Winequality red 3 vs 5	691	11	681	68.1
Ecoli 0-2-3-4 vs 5	202	7	182	9.1	Glass 0-1-6 vs 5	184	9	175	19.44	Abalone 20 vs 8-9-10	1916	8	1890	72.69
Glass 0-1-5 vs 2	172	9	155	9.12	Hypothyroid	3163	25	3012	19.95	Kddcup buffer	2233	31	2203	73.43
Yeast 0-3-5-9 vs 7-8	506	10	456	9.12	Shuttle c2 vs c4	129	9	123	20.5	Kddcup land vs satan	1610	30	1589	75.67
Yeast 0-2-5-6 vs 3-7-8-9	1004	10	905	9.14	Shuttle 6 vs 2-3	230	9	220	22	Kr-vs-K zero vs fifteen	2193	6	2166	80.22

Yeast 0-2-5-7-9 Vs 3-6-8	1004	10	905	9.14	Yeast 1-4-5-8 vs 7	693	10	663	22.1	Poker 8-9 vs 5	2075	25	2050	82
Ecoli 0-4-6 vs 5	203	6	183	9.15	Glass5	214	9	205	22.78	Poker 8 vs 6	1477	25	1460	85.88
CM1	498	23	449	9.16	Yeast 2 vs 8	482	10	462	23.1	Kddcup rootkit	2225	47	2203	100.14
										Abalone19	4174	9	4142	129.44

N, Number of samples; d, Number of variables; +N, Number of majority samples; IR, Imbalance ratio

Before applying the resampling methods, the researchers completed some feature encoding steps to make datasets applicable to resampling methods and classification algorithms. The categorical values were one-hot encoded when the values had less than 5 categories. Otherwise the values were kept originally to keep the number of variables relatively low.

5.2 Data Complexity Measure

In this study, for representation of the data characteristics, *'complexity measure'* was used. The R package Extended Complexity Library (ECoL) provides a measure to characterize the complexity of classification problems based on aspects that quantify the data linearity, the presence of informative features and the sparsity and dimensionality. Among these, this paper used the relevant measures which were feature overlapping, neighborhood and class imbalance measures.

1) The feature overlapping measures characterized how informative the available features are in separating the classes. If there was one very discriminative feature, the problem could be considered simpler than if there was no such attribute.

2) The neighborhood measures analyzed the neighborhoods of the data points and tried to capture class overlapping and the shape of the decision boundary.

3) The balance measures captured the differences in the number of samples per class in the dataset.

Each measurement assumed the value in bounded intervals (0,1]. Also, the lower values indicated a higher complexity, while the higher values indicated a higher complexity. Table 4 shows a detailed description.

Table 4. Summary of Complexity Measure

	measure	description
overlapping	F1	Maximum Fisher's Discriminant Ratio. measures the overlap between the values of the features and takes the value of the largest discriminant ratio among all the available features
	F1v	Directional-vector maximum Fisher's discriminant ratio. complements F1 by searching for a vector able to separate two classes after the training examples have been projected into it
	F3	maximum individual feature efficiency. the ratio between the number of examples that are not in the overlapping region of two classes and the total number of examples.
neighborhood	N1	Fraction of borderline points. computes the percentage of vertexes incident to edges connecting examples of opposite classes in MST
	N2	Ratio of intra/extra class nearest neighbor distance. computes the ratio of two sums: intra-class and inter-class
imbalance	C2	Index computed for measuring class balance.

5.3 Other Setup Details

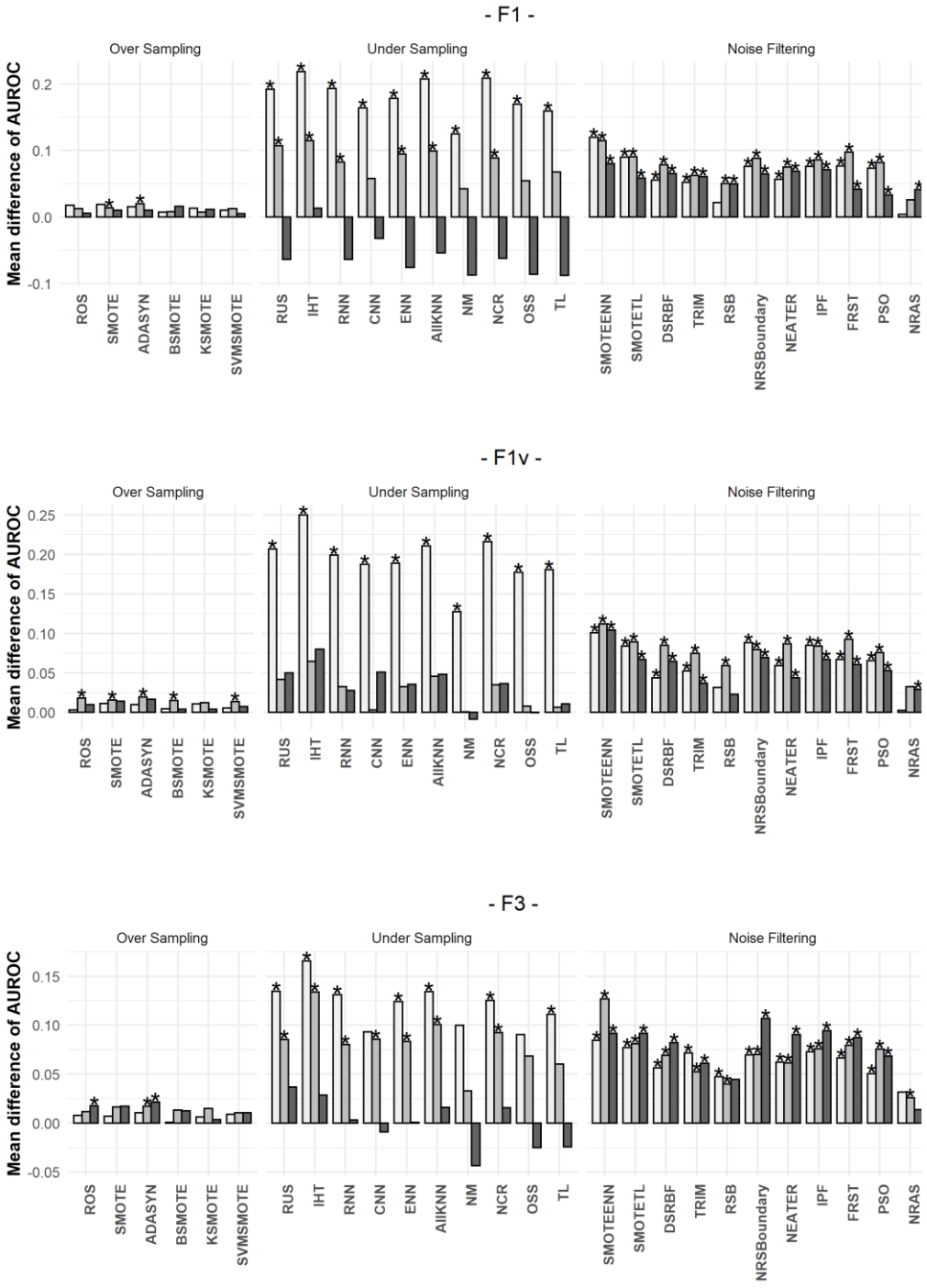
The classification methods used in this study were the decision tree (DT) classifier, K-Nearest Neighbor (KNN), Linear SVM (SVM), Random Forest (RF) and neural network (NN) to search for the best combination of classifiers and resampling methods. Training and testing sets were divided into a 7:3 ratio and optimal parameters for each classification algorithm were set using a 3-fold cross validation. For evaluation, the AUROC and AUROC were used in the same way as section 4.

5.4 Results

5.4.1 Results of Mean Difference in Performance after Resampling

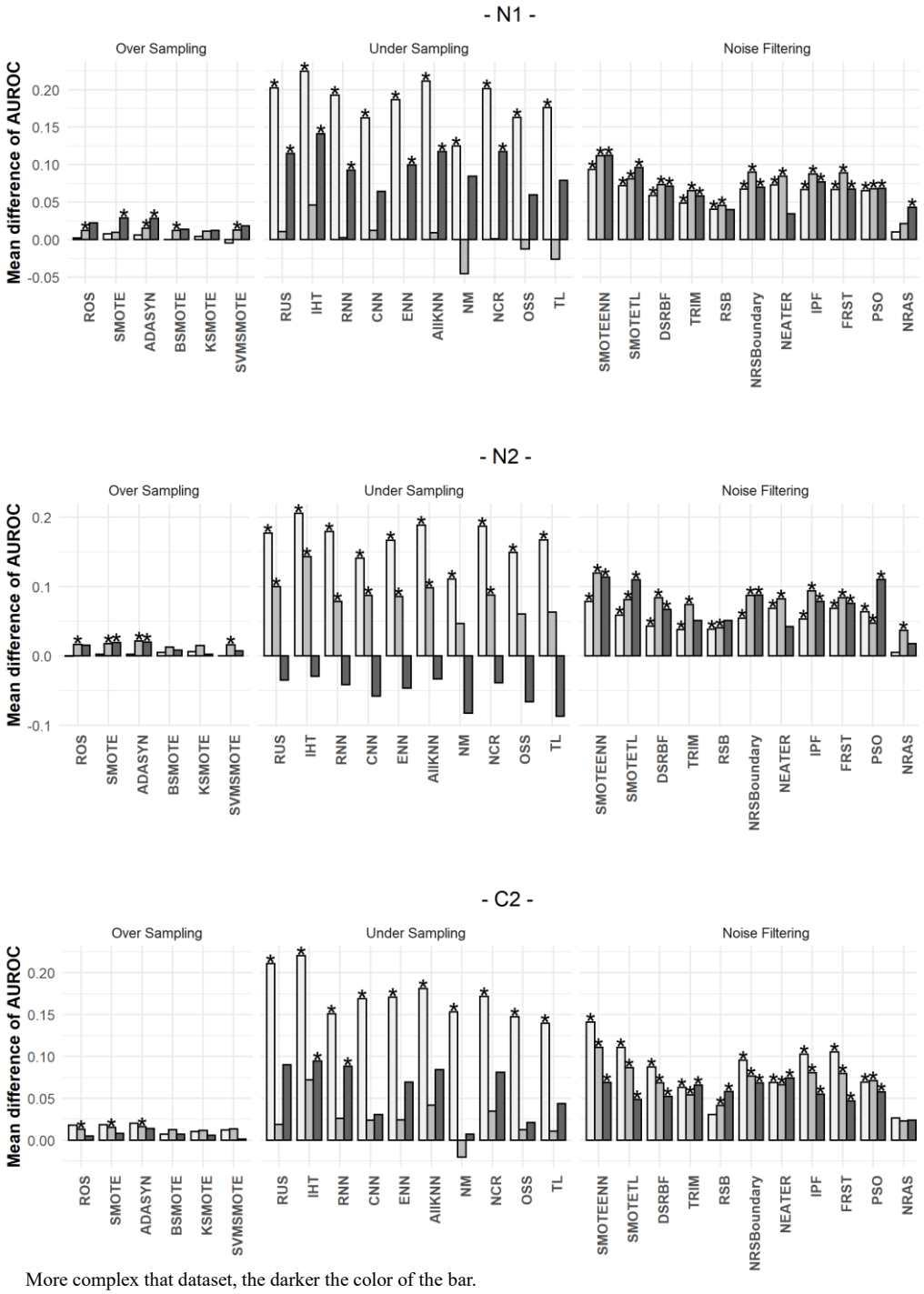
In Figure 10-13, the mean differences of original classification and result after resampling applied are shown. In order to find out the difference according to data complexity, each dataset was divided into 3 intervals using complexity measure. The more complex that dataset, the darker the color of the bar. If the value was positive, this indicated performance improvement, and if the value was negative, this indicated performance decrement. Figure 10 and 11 is when using AUROC, and Figure 12 and 13 is when using AUPRC.

Different methods were used to measure the complexity of the data, but the data showed similar results. First, when the oversampling method was applied, there were little to no increase in the AUROC in almost all complexities. This was consistent with the results of simulation in section 4. From this result, the researchers concluded that the overgeneralization problem of oversampling methods needed to be complemented by filters.



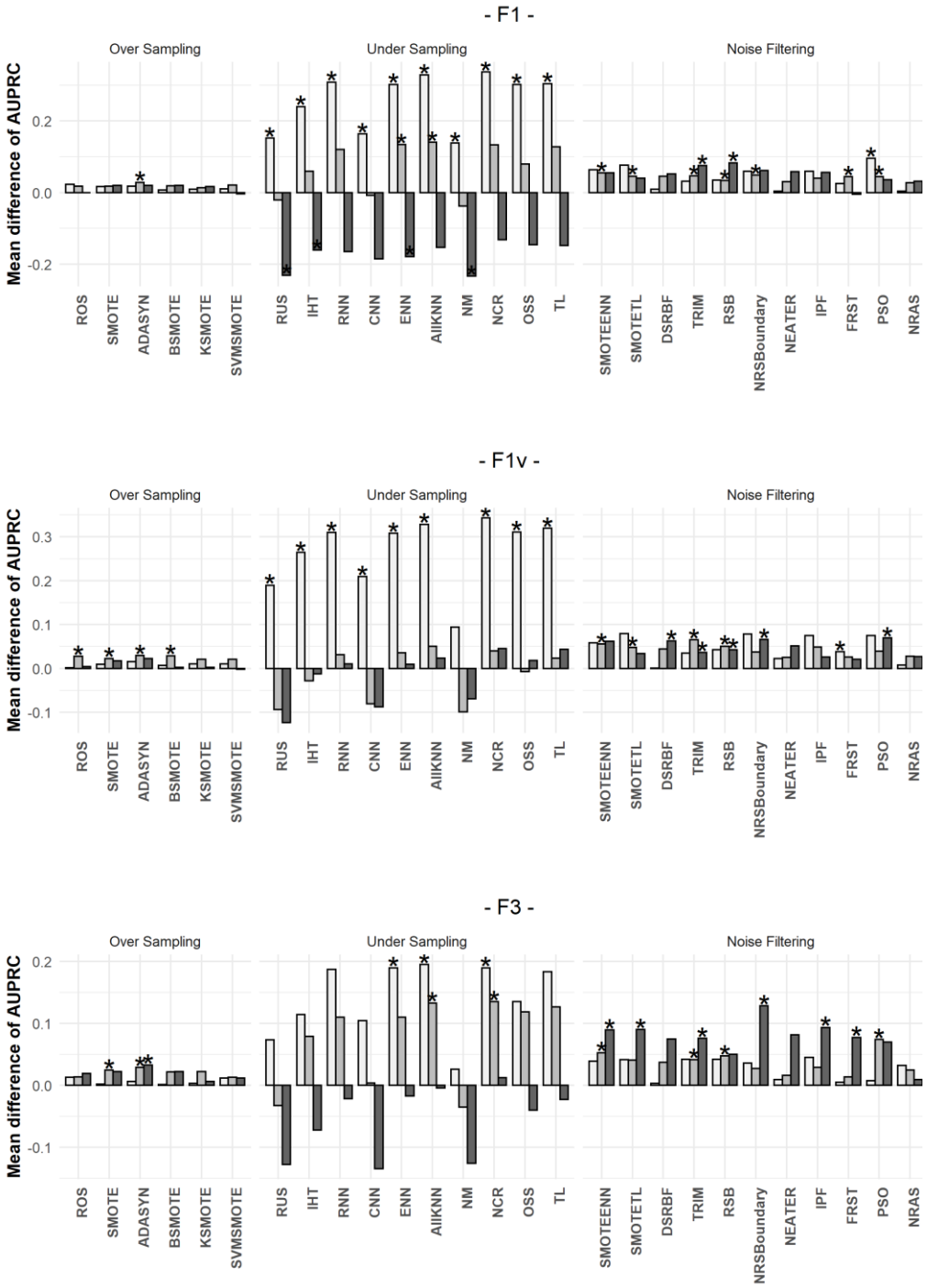
More complex that dataset, the darker the color of the bar.

Figure 10. Mean Difference of AUROC using F1, F1v and F3



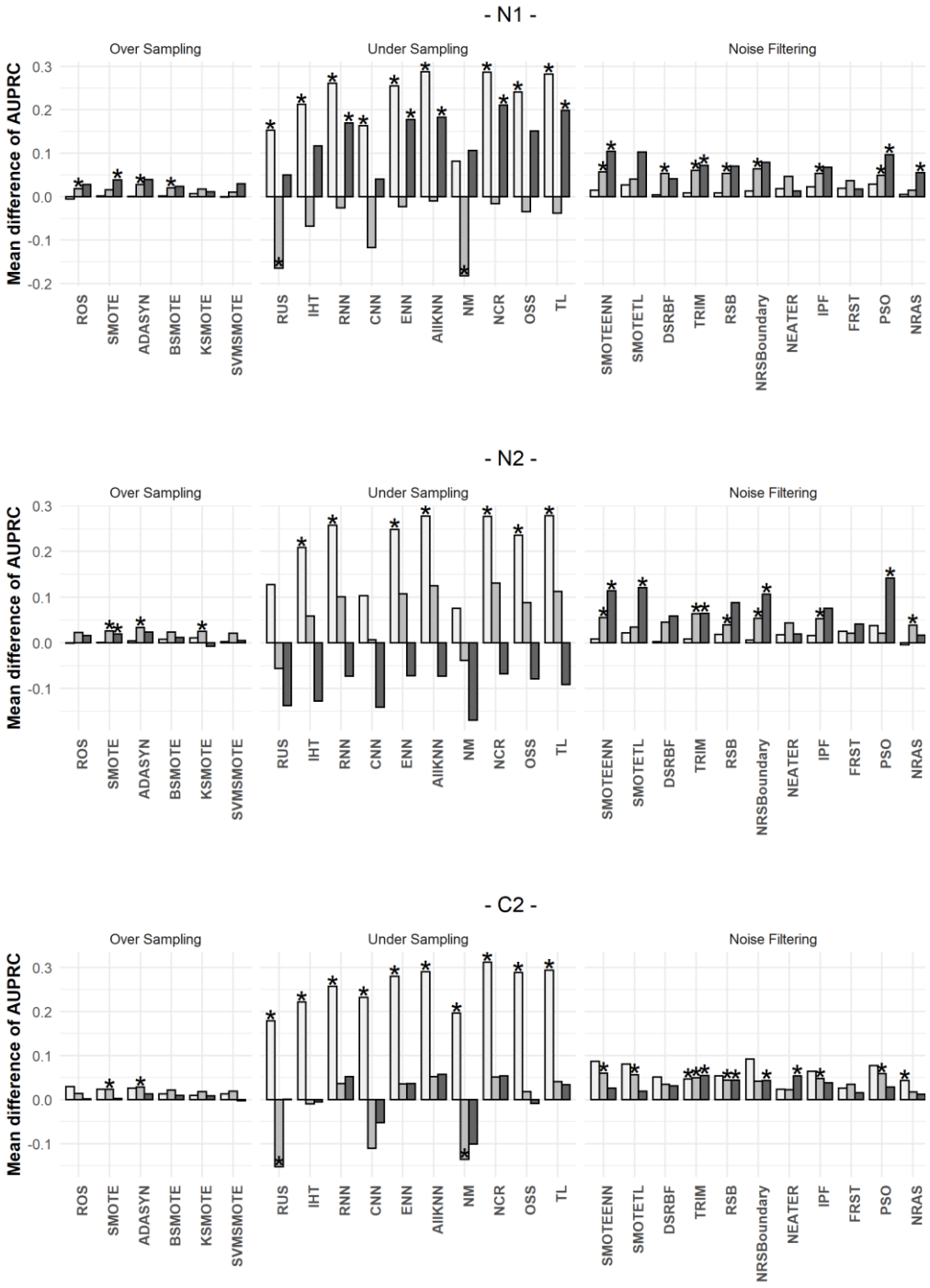
More complex that dataset, the darker the color of the bar.

Figure 11. Mean Difference of AUROC using N1, N2 and C2



More complex that dataset, the darker the color of the bar.

Figure 12. Mean Difference of AUPRC using F1, F1v and F3



More complex that dataset, the darker the color of the bar.

Figure 13. Mean difference of AUPRC using N1, N2 and C2

In the results of undersampling, when the complexity was low, the AUROC or AUPRC increased significantly. However, as the complexity increased, the effect of undersampling became insignificant. The researchers could conclude that the information loss through undersampling was small in simple datasets. In such datasets, performance could be improved only by matching the balance of two classes.

Conversely, the filtering method showed a different pattern. In a relatively simple dataset, the mean increase was less than that of undersampling. However, unlike undersampling, the filtering method showed stable results in overall complexities.

5.4.2 Results of top 10 Resampling and Classifier Combinations

Table 5-8 showed the top 10 performance combinations by applying all the resampling and classifiers to the given data to search for the optimal combination in imbalanced and complex datasets. To compare the result, the real data were divided into *'complex'* and *'non-complex'* cases.

In the process, complexity measures were used to find complex and non-complex cases among given datasets. First, C2 was used to find imbalanced data to which resampling should be applied. This paper specified imbalanced data as the C2 value in the top 25% or higher in all datasets. Next, F3 and N2, which were well-distributed values among all complexity measures, were used to classify complex and non-complex cases. Through each measure, the value of the top 25% was defined as complex case, and the value of the bottom 25% was defined as non-complex case. In Figure 14, the parts colored in light grey are an imbalanced and non-complex area, and the parts colored in dark gray are an imbalanced and complex area.

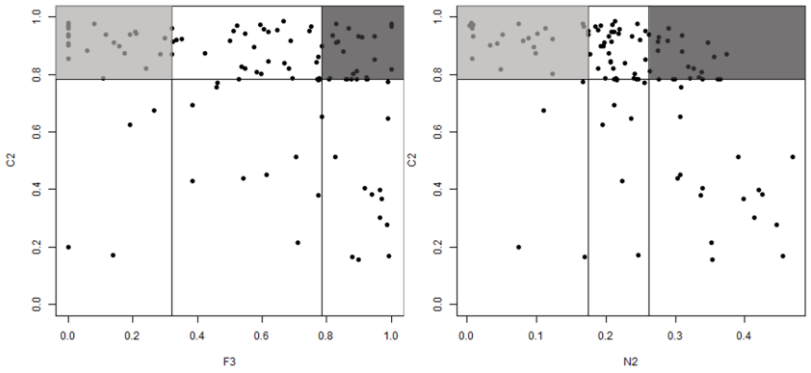


Figure 14. Complex and Non-Complex Area in Real Datasets

Table 5. Top 10 Combinations in Non-Complex Datasets (F3)

Rank	Resampling	Classifier	Average Rank	AUROC Rank
1	SMOTE-IPF	RF	33.63	2
2	SMOTE-FRST-2T	RF	34.90	1
3	NRSBoundary	RF	36.13	4
4	DSRBF	RF	40.04	7
5	SMOTE-RSB*	RF	40.65	6
6	NRAS	RF	41.42	14
7	NCR	RF	42.94	15
8	SMOTE-PSO	RF	43.19	3
9	SMOTE	RF	43.63	24
10	Borderline SMOTE	RF	43.96	17

Light gray, undersampling; Dark gray, oversampling; non-colored, filtering

Of the 140 combinations, only the top 10 combinations are shown.

Table 6. Top 10 Combinations in Non-Complex Datasets (N2)

Rank	Resampling	Classifier	Average Rank	AUROC Rank
1	SMOTE-IPF	RF	30.13	1
2	NRSBoundary	RF	35.69	2
3	DSRBF	RF	36.56	6
4	ALL KNN	RF	39.83	10
5	SMOTE-FRST 2T	RF	40.33	5
6	SMOTE-TL	RF	41.85	9
7	SMOTE	RF	42.92	14
8	SVM SMOTE	RF	43.13	15
9	**none**	RF	43.94	24
10	ALL KNN	DT	44.21	19

Light gray, undersampling; Dark gray, oversampling; non-colored, filtering; none, no resampling is applied;

Of the 140 combinations, only the top 10 combinations are shown.

Table 7. Top 10 Combinations in Complex Datasets (F3)

Rank	Resampling	Classifier	Average Rank	AUROC Rank
1	SMOTE-PSO	DT	35.92	2
2	SMOTE	DT	38.08	35
3	NRAS	DT	39.10	5
4	SMOTE-TL	RF	39.12	15
5	SMOTE-IPF	RF	39.66	19
6	Random Oversampling	DT	40.16	28
7	ADASYN	DT	40.36	41
8	SMOTE-TL	DT	40.52	4
9	AMSCO	DT	42.44	1
10	NRSBoundary	RF	42.56	20

Dark gray, oversampling; non-colored, filtering

Of the 140 combinations, only the top 10 combinations are shown.

Table 8. Top 10 combinations in Complex Datasets (N2)

Rank	Resampling	Classifier	Average Rank	AUROC Rank
1	SMOTE-TL	RF	35.00	8
2	SMOTE-IPF	RF	36.43	18
3	SMOTE-RSB*	RF	37.00	45
4	DSRBF	RF	38.64	30
5	SMOTE-FRST 2T	RF	41.00	20
6	NRSBoundary	RF	43.59	25
7	SMOTE-PSO	DT	45.16	31
8	SMOTE-PSO	RF	46.84	16
9	NCR	RF	46.91	63
10	SMOTE	DT	46.93	40

Light gray, undersampling; Dark gray, oversampling; Non-colored, filtering

Of the 140 combinations, only the top 10 combinations are shown.

Table 5-8 showed the top 10 combinations of resampling and classifier with highest average ranking based on the AUPRC. Table 5 and 7 showed the result of non-complex and complex cases based on F3. Table 6 and 8 showed the result of non-complex and complex cases using N2. In each table, the average ranking, ranking based on the AUROC were shown as well.

First, looking at the entire tables, all classifiers with high ranking were the tree based algorithms such as random forest or decision tree classifier. Through this, it could be inferred that the tree based algorithm is a strong method to imbalanced datasets. As for the resampling method, the filtering methods occupied the top ranking regardless of the complexity of the datasets. The combination of RF+SMOTE IPF and RF+NRSBoundary showed high ranking in all cases. In addition, although it has a lower ranking compared to

the filtering methods, SMOTE, an oversampling method, also occupied a high rank. This can be inferred that the selection of the classifier has a greater effect on the classification performance.

Tables 7 and 8, which were more complex cases, an undersampling occupied relatively lower ranks and the filtering methods occupied higher ranks. This result was consistent with the results of section 4. In Table 7, there were no undersampling in the top 10 combinations, and in Table 8, there were one undersampling, but it ranked 9th. Among the filtering methods, SMOTE-TL was suitable when F3 was high, and SMOTE-PSO was suitable when N2 was high.

Tables 5 and 6, which were non-complex cases, showed that the ranks of undersampling were relatively high compared to the complex cases similar to the results of section 4. In particular, when N2 was low, ALL KNN ranked high with the combination with random forest and decision tree classifier. However, even when the complexities were low, the filtering methods occupied high ranks. Specifically, RF+SMOTE-FRST-2T and RF+DSRBF showed high average rank in non-complex cases.

In these tables, it was unusual that oversampling ranked high in both complex and non-complex cases. However, oversampling methods with different features were selected according to the complexity. In case of complex datasets, the algorithm that does not control the region to be oversampling occupied a high rank, whereas in the non-complex case, the algorithm that controls the region occupied high ranks. Also, in Table 6, the case where resampling was not applied also occupied a high rank. It can be inferred that, if the data is not complex, the selection of the right classifier could solve class imbalance problems.

6. Conclusion

The objective of this paper is to provide an optimal resampling method for complex datasets. In several studies, oversampling showed poor classification results in imbalanced settings. This paper suggested two alternative approaches to overcome this problem in complex datasets.

Through various simulation scenarios, the researchers saw that applying oversampling showed poor performance overall due to overgeneralization problems. This paper suggested applying an undersampling or filtering method instead. In case of non-complex datasets, undersampling was found to be optimal. However, performance vary according to the degree of imbalance. In case of complex datasets, applying a filtering method to delete misallocated examples found to be optimal.

Based on the real data analysis, the best combinations of classifier and resampling methods for each data characteristic were provided. Overall, the combination of random forest classifier and filtering methods showed highest performance. SMOTE-IPF+RF and NRSBoundary+RF showed high ranking in all four cases provided. In complex and imbalanced cases, in addition to the aforementioned filtering methods, SMOTE-TL and SMOTE-PSO showed high ranking in combination with tree based classifiers. In non-complex and imbalanced cases, the combination of ALL KNN, an undersampling technique, and tree based classifier showed high performance.

Research on resampling methods has opened up new opportunities for improving

classification performance on imbalanced datasets. Many researchers have contributed to the development of the resampling method because of its convenient and versatile features. However, the empirical behavior of the resampling method is highly dependent on the observed data characteristics. The researchers believe that this study can contribute to improvement of classification performance in imbalanced and complex datasets and the further development of resampling methods.

References

Ho, T. K., & Basu, M. (2002). Complexity measures of supervised classification problems. IEEE transactions on pattern analysis and machine intelligence, 24(3), 289-300.

Zhu, X., & Wu, X. (2004). Class noise vs. attribute noise: A quantitative study. Artificial intelligence review, 22(3), 177-210.

Park, G. U., & Jung, I. (2019). Comparison of resampling methods for dealing with imbalanced data in binary classification problem. The Korean Journal of Applied Statistics, 32(3), 349-374.

Batista, G. E., Prati, R. C., & Monard, M. C. (2004). A study of the behavior of several methods for balancing machine learning training data. ACM SIGKDD explorations newsletter, 6(1), 20-29.

Fernández-Navarro, F., Hervás-Martínez, C., & Gutiérrez, P. A. (2011). A dynamic over-sampling procedure based on sensitivity for multi-class problems. Pattern Recognition, 44(8), 1821-1833.

Puntumapon, K., & Waiyamai, K. (2012, May). A pruning-based approach for searching precise and generalized region for synthetic minority over-sampling. In Pacific-Asia Conference on Knowledge Discovery and Data Mining (pp. 371-382). Springer, Berlin, Heidelberg.

Ramentol, E., Caballero, Y., Bello, R., & Herrera, F. (2012). SMOTE-RSB*: a hybrid preprocessing approach based on oversampling and undersampling for high imbalanced data-sets using SMOTE and rough sets theory. Knowledge and information systems, 33(2), 245-265.

Hu, F., & Li, H. (2013). A novel boundary oversampling algorithm based on neighborhood rough set model: NRSBoundary-SMOTE. Mathematical Problems in Engineering, 2013.

Almogahed, B. A., & Kakadiaris, I. A. (2015). NEATER: filtering of over-sampled data using non-cooperative game theory. Soft Computing, 19(11), 3301-3322.

Sáez, J. A., Luengo, J., Stefanowski, J., & Herrera, F. (2015). SMOTE–IPF: Addressing the noisy and borderline examples problem in imbalanced classification by a re-sampling method with filtering. Information Sciences, 291, 184-203.

Ramentol, E., Gondres, I., Lajes, S., Bello, R., Caballero, Y., Cornelis, C., & Herrera, F. (2016). Fuzzy-rough imbalanced learning for the diagnosis of High Voltage Circuit

Breaker maintenance: The SMOTE-FRST-2T algorithm. Engineering Applications of Artificial Intelligence, 48, 134-139.

Rivera, W. A. (2017). Noise reduction a priori synthetic over-sampling for class imbalanced data sets. Information Sciences, 408, 146-161.

Japkowicz, N., & Stephen, S. (2002). The class imbalance problem: A systematic study. Intelligent data analysis, 6(5), 429-449.

Dua, D. and Graff, C. (2019). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

Luis Garcia and Ana Lorena (2019). ECoL: Complexity Measures for Supervised Problems. https://CRAN.R-project.org/package=ECoL

국 문 요 약

복잡도가 높은 불균형 자료에서

최적의 표본 재추출 방법

대부분의 학습 알고리즘은 계급 간 균형을 가정한다. 계급 간 불균형한 자료의 경우 분류 알고리즘은 다수 계급에 쉽게 편향되기에 이는 중요한 문제이다. 계급 불균형을 해결하기 위한 방법은 크게 데이터 수준과 알고리즘 수준에서의 해결책으로 나뉜다. 데이터 수준에서의 해결책 즉 표본 재추출 방법은 다양한 분류 알고리즘과 함께 사용이 가능하다는 점에서 많은 연구에서 사용되고 새로운 방법들도 개발되고 있다. 하지만 몇몇 연구 결과에서는 가장 많이 사용되는 표본 재추출 방법인 오버샘플링이 분류 성능을 악화시키는 것을 보인다. 이는 오버샘플링의 과잉 일반화 문제 때문이다. 과잉 일반화 문제는 오버샘플링으로 생성된 임의 데이터가 소수 계급을 대표해야 하지만 다수 계급 공간에 생성된 경우를 말한다. 본 논문에서는 과잉 일반화 문제는 복잡

한 자료에서 악화한다고 주장한다. 본 연구는 복잡한 자료에서 과잉 일반화 문제를 해결하기 위한 두 가지 방법을 제안한다. 첫 번째 방법은 오버샘플링에 필터를 결합하는 것, 두 번째는 언더샘플링을 적용하는 것이다. 본 연구의 목적은 복잡한 자료에서 최적의 표본 재추출 방법을 제안하는 것이다. 이를 위해 다양한 복잡성, 불균형 정도, 표본 크기를 고려하는 모의실험과 실제 자료를 이용해 표본 재추출 방법을 비교하였다. 본 연구를 통해 자료의 복잡도에 따라 적합한 표본 재추출 방법을 제시하는 바이다.

핵심되는 말: 불균형 자료, 과잉 일반화, 표본 재추출 방법, 오버 샘플링, 언더 샘플링