Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

A semi-supervised resampling method for class-imbalanced learning

Zhen Jiang [*], Lingyun Zhao, Yu Lu, Yongzhao Zhan, Qirong Mao

School of Computer Science and Communication Engineering, JiangSu University, ZhenJiang, China

A R T I C L E I N F O

Dataset link: https://codeocean.com/capsule/5051082/tree/v1

Keywords:
Class-imbalanced learning
Hybrid resampling
Semi-supervised hierarchical clustering
Cluster-splitting

A B S T R A C T

Clustering analysis is widely used as a pre-process to discover the data distribution for resampling. Existing clustering-based resampling methods mostly run unsupervised clustering on labeled data without taking advantage of the class information to guide the clustering. When there are not enough labeled data, the clustering can hardly capture the underlying data distribution. In this paper, we propose a semi-supervised hybrid resampling (SSHR) method which runs semi-supervised clustering to capture the data distribution for both over-sampling and under-sampling. Firstly, we design a semi-supervised hierarchical clustering algorithm (SSHC) which uses labeled data to guide the clustering procedure on the whole dataset. Specifically, labeled data are used to initialize a clustering model and then guide its updating via an iterative cluster-splitting process. In this way, original classes are divided into multiple disjunct clusters, which contributes o disclosing not only the inter-class imbalance but also the intra-class imbalance. Subsequently, a hybrid resampling is performed according to the result of SSHC Labeled data of the majority class are under-sampled according to the distances to their cluster centroids and the adjacency to minority cluster centroids. Furthermore, we propose a novel over-sampling approach which selects some confident unlabeled data in minority clusters as pseudo-labeled data to enlarge the training set Compared with traditional over-sampling methods, our approach contributes to discovering more about the distribution of the minority class. In order to validate the effectiveness of SSHR, we conduct extensive experiments on 44 benchmark datasets. Our method achieves the best performances in terms of both F-measure and AUC. The Friedman test demonstrates that SSHR significantly outperforms the compared state-of-the-art resampling algorithms.

1. Introduction

Learning from datasets with imbalanced class distributions remains a challenge in machine learning. Since traditional classification algorithms are designed to minimize the overall prediction errors, the classifier can be biased towards majority classes and thereby perform poorly on minority classes (Kaur, Pannu, & Malhi, 2019). Unfortunately, minority classes are usually valuable in real applications, e.g., medical diagnosis, fraud detection, and many others. Over the past decades, numerous techniques have been proposed for class-imbalanced learning (CIL), which can be roughly grouped into algorithmic-level and data-level methods.

Algorithmic-level methods modify the preference criteria of a selected classification algorithm (Elkan, 2001; Iranmehr, Masnadi-Shirazi, & Vasconcelos, 2019) or incorporate benefits and/or costs into the classification algorithm (Mullick, Datta, & Das, 2018; Richhariya & Tanveer, 2020). It is noteworthy that this branch needs a deep knowledge of both the selected classification algorithm and the related domain. Also, the work in Weiss (2004) points out that cost-sensitive learning may cause an over-fitting problem. Data-level methods pre-process imbalanced datasets to make them suitable for standard classification algorithms. The most popular approach is to oversample the minority class or/and undersample the majority class until a required degree of balance is reached (Chawla, Bowyer, Hall, & Kegelmeyer, 2002; Kubat, Matwin, et al., 1997). However, deciding the optimal distribution is not straightforward. Moreover, oversampling is liable to bring noise, while undersampling has the danger of losing important information (Kaur et al., 2019).

A variety of techniques have been proposed to improve the effect of resampling. Recently, clustering analysis has received significant attention due to its natural suitability to discover the data distribution for resampling (Lu et al., 2022). Generally, these methods separately assign majority or/and minority samples to a number of clusters. Then oversampling or/and undersampling is performed within these clusters to achieve a class balance. Note that such a local clustering can hardly discover the global data distribution. Moreover, existing clustering-based resampling methods mostly run unsupervised clustering on labeled data. Without taking advantage of the class information,

* Corresponding author.
E-mail addresses: jiangz@ujs.edu.cn (Z. Jiang), 2222008096@stmail.ujs.edu.cn (L. Zhao), 2211908021@stmail.ujs.edu.cn (Y. Lu), yzzhan@ujs.edu.cn (Y. Zhan), mao_qr@ujs.edu.cn (Q. Mao).

https://doi.org/10.1016/j.eswa.2023.119733
Received 5 July 2022; Received in revised form 20 February 2023; Accepted 20 February 2023
Available online 24 February 2023

the clustering is difficult to find an optimum solution for resampling and the following classification. For example, it remains a challenging issue to determine how many clusters should exist for Kmeans, which is used in most clustering-based methods. Significantly, the clustering can hardly disclose the underlying distribution when there is not enough labeled data. On the other hand, semi-supervised clustering (Van Engelen & Hoos, 2020) efficiently improves the clustering performance with the support of prior knowledge (usually obtained from labeled data). Nevertheless, leveraging the strength of semi-supervised clustering is still an underexplored field in CIL with few solutions (Kaur et al., 2019; Nekooeimehr & Lai-Yuen, 2016).

In this paper, we present a hybrid resampling method that runs semi-supervised clustering on both labeled and unlabeled data to discover the data distribution for the resampling. Especially, we propose a novel semi-supervised hierarchical clustering algorithm, SSHC, which splits initial classes into a number of clusters with the supervision of labeled data. Based on the "compact-cluster" assumption (Jiang, Zhan, Mao, & Du, 2022), any "impure" cluster that contains labeled data from different classes is split into multiple "pure" sub-clusters. This way, SSHC can efficiently disclose both inter-class and intra-class imbalances in the data distribution to facilitate the following resampling. For undersampling, we first delete majority samples in minority clusters and small-size majority clusters, which are usually considered noises and outliers. Subsequently, majority samples are deleted according to the distance to their centroids and the distance to nearby minority clusters. In addition to alleviating the bias towards the majority class, these approaches also contribute to addressing the class-overlapping issue. For oversampling, we select unlabeled data close to centroids of minority clusters as minority samples since the data in one cluster are likely to share the same label. In contrast to repeatedly adding original samples or generating synthetic samples, the proposed oversampling method effectively alleviates the deficiency of labeled data. It captures more details about the actual distribution of the minority class. By modifying the class distribution with a semi-supervised resampling, the proposed method can be readily applied to any existing classification algorithms. It has a wide application prospect in class-imbalanced tasks, such as medical diagnosis, fraud detection, text classification, etc.

The proposed method is compared with nine state-of-the-art resampling methods on 44 real-world datasets. The source code and related datasets are available at https://codeocean.com/capsule/5051082/tree/v1. The extensive experimental results demonstrate that SSHR has significant superiority over the compared methods. The contributions of this paper are highlighted as follows.

- We present a hybrid resampling method that introduces semi-supervised clustering into CIL for resampling. By clustering both labeled and unlabeled data, the proposed method can efficiently capture the underlying data distribution to facilitate the following resampling.
- Based on the "compact-cluster" assumption, we propose a novel semi-supervised hierarchical clustering algorithm, SSHC, which iteratively splits initial classes into a number of disjunct clusters with the guide of labeled data. In this way, the proposed method can identify the "small disjuncts" (Jo & Japkowicz, 2004) within classes and simultaneously handle both inter-class and intra-class issues.
- We propose an oversampling approach which uses unlabeled data in minority clusters to enlarge the training set. Compared with traditional oversampling methods, it can disclose more details about the data distribution and alleviate the deficiency of labeled data in many real-world tasks.
- We conducted extensive experiments on 44 real-world datasets to show the effectiveness of the proposed resampling method. The result demonstrates that SSHR can improve the performance of CIL algorithms.

The rest of this paper is organized as follows: Section 2 reviews related works in resampling methods and semi-supervised clustering. After illustrating the motivation, Section 3 presents the SSHC algorithm and the hybrid resampling method, together with a related complexity analysis. The experimental evaluations are reported in Section 4. Finally, we conclude the paper and describe possible future work in Section 5.

2. Related work

2.1. Resampling methods for CIL

Resampling aims to rebalance the dataset by either oversampling the minority class or undersampling the majority class so that any standard learning algorithm can be applied to the pre-processed dataset.

2.1.1. Oversampling methods

Oversampling methods increase the number of minority samples to achieve some degree of class balance. The straightforward approach is to add samples from the original dataset. However, it increases the likelihood of overfitting. An alternative solution is to generate synthetic samples for minority classes. Zhang and Li introduced perturbations via random-walk to generate new minority samples (Zhang & Li, 2014). The best-known SMOTE algorithm (Chawla et al., 2002) generates minority samples by linearly interpolating between a randomly selected minority sample and its k nearest minority neighbors. However, this approach may produce some noisy samples due to its sensitivity to k value and no consideration of the majority class while generating new samples. Numerous variants of SMOTE have been proposed to address this problem. The first type of SMOTE variant removes some "bad" synthetic samples with a post-processing mechanism, such as Tomek Links (Batista, Prati, & Monard, 2004) and Edited Nearest Neighbors (Bej, Davtyan, Wolfien, Nassar, & Wolkenhauer, 2021). The second type introduces some modifications in the way SMOTE generates the synthetic samples, e.g., generates synthetic samples according to density (Bej et al., 2021; Bunkhumpornpat, Sinapiromsaran, & Lursinsap, 2012) or the distance to seed (Bunkhumpornpat, Sinapiromsaran, & Lursinsap, 2009). The third type only generates synthetic samples in specific regions that are considered helpful for the learning algorithm, such as the borders between classes (Han, Wang, & Mao, 2005) or those harder-to-learn samples (Barua, Islam, Yao, & Murase, 2012; He, Bai, Garcia, & Li, 2008).

Due to the flexible ability to disclose the data structure, clustering analysis is widely used in resampling methods. To deal with small disjuncts, Jo and Japkowicz (2004) separately clustered each class and oversampled the sub-clusters of the same class to ensure they have equal sizes. Similarly, Douzas et al. divided the dataset into several clusters, each of which is oversampled according to its imbalance rate (IR) and density (Douzas, Bacao, & Last, 2018). Instead of clustering all classes, other methods only cluster the minority class and perform oversampling within these clusters. The synthetic samples can be generated according to the clusters' centers (Douzas & Bacao, 2019) or the clusters' classification complexity (Nekooeimehr & Lai-Yuen, 2016). Unlike previous methods which use Kmeans for the clustering, Tao et al. applied a density peaks clustering to cluster the minority samples. They oversampled each cluster according to its size and density (Tao et al., 2020).

Recently, Generative Adversarial Neural Networks (GAN) has been applied to synthesize minority samples to handle the imbalance problem (Elyan, Jamieson, & Ali-Gombe, 2020; Engelmann & Lessmann, 2021; Hao et al., 2021). The methods have shown favorable performance on image datasets over traditional resampling techniques.

2.1.2. Undersampling methods

Undersampling removes the majority samples from the original dataset to decrease the dominance of the majority class. A diverse set of techniques have been proposed to avoid the loss of meaningful information. These methods can be categorized into two groups:

(1) neighborhood-based methods remove the majority samples based on k-nearest neighbors. Tomek Ivan first proposed a hybrid form of Condensed Nearest Neighbour Rule (CNN) and Tomek-Link to undersampling data (Tomek, 1976). The NUS (Kang, Chen, Li, & Zhou, 2016) combines the advantages of noise filter and under-sampling to achieve better classification performance. Moreover, the evolutionary algorithm was incorporated as a strategy to perform under-sampling through a prototype selection procedure (García & Herrera, 2009). Recently, Vuttipittayamongkol and Elyan (2020) proposed an under-sampling framework to eliminate the class overlapping. This framework maximizes the visibility of minority samples and reduces the excessive removal of samples.

(2) Clustering-based methods aim to code the data structure before sampling the majority class. Yen and Lee (2009) used Kmeans to cluster all training samples and then randomly removed a number of majority samples from each cluster by considering its imbalance ratio. The Cluster-NN (Bashir, Doolan, & Petrovski, 2015) clusters only majority samples and then uses the cluster centers or their nearest neighbors to represent the majority class to keep balance with the minority class. The CBIS method (Tsai, Lin, Hu, & Yao, 2019) utilizes affinity propagation to group the majority samples into K clusters and filters out samples in each cluster to construct a balanced dataset. By contrast, the Fast-CBUS (Ofek, Rokach, Stern, & Shabtai, 2017) clusters minority samples and only selects a similar number of majority samples for each cluster. Building on the information-preservation principle, the RIUS selects the most relevant examples from the majority class. Additionally, the RIUS is coupled with the CBUS to enhance the data representation (Hoyos-Osorio, Alvarez-Meza, Daza-Santacoloma, Orozco-Gutierrez, & Castellanos-Dominguez, 2021).

2.1.3. Hybrid methods

With the idea to extract the strong points and reduce the weaknesses in a single method, several hybridization methods have been reported to have highly competitive and robust performance in the past decade. A few reported works are the hybridization of oversampling with undersampling. Ramentol, Caballero, Bello, and Herrera (2012) proposed a hybrid resampling method that uses SMOTE and rough sets to improve oversampling and undersampling, respectively. The CDBH algorithm (Mirzaei, Nikpour, & Nezamabadi-pour, 2021) uses kmeans to cluster minority samples and majority samples separately. Subsequently, oversampling and undersampling are performed simultaneously according to the density of clusters. The most popular hybrid approach is to perform resampling as a pre-processing in the context of ensemble learning. Random undersampling was firstly combined with bagging (Tao, Tang, Li, & Wu, 2006) or boosting (Kumar, Biswas, & Devi, 2019; Seiffert, Khoshgoftaar, Van Hulse, & Napolitano, 2010). A variant is to combine undersampling with boosting via Entropy (Wang, Cao, & Zhu, 2020) or Evolutionary algorithm (Galar, Fernández, & Herrera, 2013). Lin, Tsai, Hu, and Jhang (2017) used clustering analysis to undersample the majority class and then combined it with the Adaboost technique. Lim, Goh, and Tan (2016) implemented an evolutionary ensemble framework by clustering the minority samples using mini-batch kmeans and hierarchical agglomerative clustering before generating synthetic samples. Moreover, the SMOTE was also incorporated into the bagging (Sun, Lang, Fujita, & Li, 2018) or boosting context (Zhang, Ramezani, & Naeim, 2019).

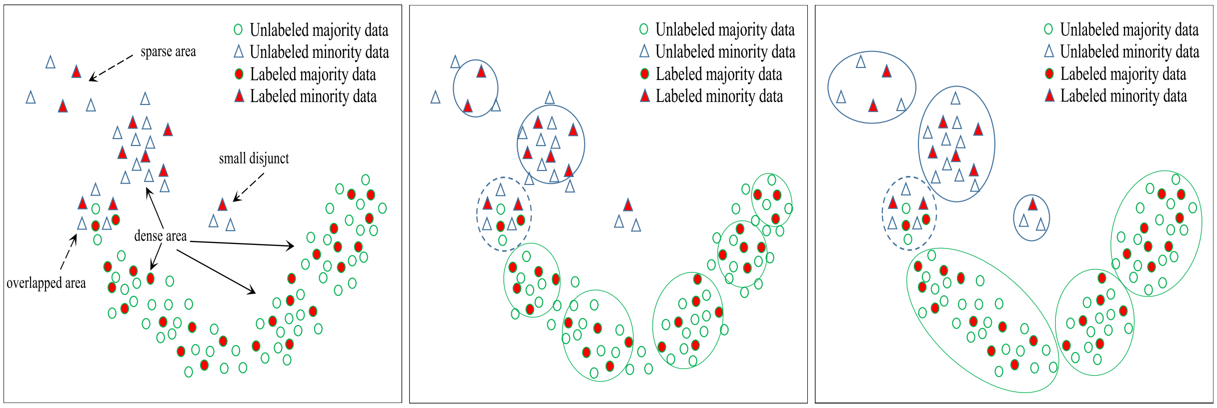
2.2. Semi-supervised clustering methods

Existing clustering-based CIL methods mostly perform unsupervised clustering (the most commonly used is Kmeans) on labeled data to discover the data structure for resampling. Without taking advantage of the class information, the unsupervised clustering can hardly disclose the actual data distribution for the resampling, especially when there are not enough labeled data. By contrast, semi-supervised clustering uses side information in the form of class labels or pairwise constraints to guide the clustering on both labeled and unlabeled data towards an optimum partition.

Constraint-based clustering methods integrate some type of constraints into original clustering objective functions to bias the search strategy towards a partition that respects these constraints as much as possible. The most common type is the pairwise constraint which specifies whether a pair of data should be grouped together (must-link) or not (cannot-link). In Kmeans-style methods (Lai et al., 2019; Wagstaff, Cardie, Rogers, Schrödl, et al., 2001), the objective functions are usually modified to ensure that the cluster memberships are consistent with given constraints. In spectral clustering algorithms, the pairwise constrains are incorporated as a regularizer to modify the affinity matrix (Xu, Desjardins, & Wagstaff, 2005) or the Laplacian graph (Chen, Qian, Chen, Zheng, & Zhu, 2019) for the following clustering. Similarly, a variation of DBSCAN (Ren et al., 2018) was proposed by incorporating pairwise constraints into density-based clustering. Moreover, the pairwise constraints are also incorporated into kernel mean shift clustering (Anand, Mittal, Tuzel, & Meer, 2013) to guide the clustering procedure. It is noteworthy that the clustering performance suffers from the vulnerability of the order of constraints.

Label-based clustering uses a part of labeled data to supervise the clustering procedure. Compared with pairwise constraints, the partial labeling information is more natural to describe data at a high level, and it is immune to the impact of the order of side information. In Kmeans-style methods, the labeled data are used to initialize seeds or further constrain the centroid-updating process (Basu, Banerjee, & Mooney, 2002; Yoder & Priebe, 2017). In a similar way, the labeled data are used to estimate the distinct local density parameters in density-based algorithms (Gertrudes, Zimek, Sander, & Campello, 2018; Vu & Do, 2017). Another approach is adding a regularization term into the objective function to penalize the partition inconsistent with the given label information (Zeng, Tong, Sang, & Huang, 2013). Liu et al. proposed a Partition Level Constrained Clustering framework (Liu, Tao, & Fu, 2017) where the class labels are added as additional dimensions to original feature vectors. It provided solutions for Kmeans and spectral clustering, respectively. Most recently, Jiang et al. presented Compact-cluster-based SSC framework (CSSC) which takes advantage of labeled data to seek the optimum number of clusters and their centroids (Jiang et al., 2022).

Despite the substantial ability to capture data distribution, semi-supervised clustering is rarely applied to CIL. Existing clustering-based methods mostly run unsupervised clustering on labeled data without utilizing the label information to disclose the class distribution. An exception is the adaptive semi-unsupervised weighted oversampling (A-SUWO) method (Nekooeimehr & Lai-Yuen, 2016) which proposes a semi-unsupervised hierarchical clustering algorithm. In a bottom-up clustering process, the class information of majority samples is utilized to merge minority sub-clusters and then synthesize informative and non-overlapping samples. Nevertheless, leveraging the strength of unlabeled data is still an underexplored field in CIL. It is beneficial when the number of labeled data is insufficient to reveal the data distribution. Based on the compact cluster assumption (Jiang et al., 2022), we propose a semi-supervised hierarchical clustering algorithm that firstly utilizes unlabeled data to disclose the class distribution for resampling. According to the result of SSHC, unlabeled data is further utilized for oversampling, which is another novelty point of the proposed method.



(a) Initial class distribution (b) The class distribution disclosed by unsupervised clustering on labeled data (c) The class distribution disclosed by semi-supervised hierarchical clustering

Fig. 1. The effectiveness of semi-supervised hierarchical clustering on CIL.

Algorithm 1: The SSHC algorithm

```
Input: $D_L, D_U, P = \{\Phi\}$
1. For each class $y_i$ in $Y$
      Add an empty cluster $p_i$ with label $y_i$ to $P$
      Assign samples of class $y_i$ to $p_i$ and compute the centroid $m_i$ using Eq. (1)
2. Repeat until no new clusters appear
      Assign each $x_i \in X_L \cup X_U$ to its closest cluster
      For each $p_h \in P$ that contains samples from multiple classes
         For each class $j$ whose label is different from $p_h$
            Assign $\forall x \in p_h$ with label $j$ to a new cluster $p_h^j$
            Compute the centroid of $p_h^j$ using Eq. (1)
      Reassign each $x_i \in X_L \cup X_U$ to its closest cluster
      Compute the objective value of the current clustering model using Eq. (3)
      If the objective value decreases
         Restore the former clustering model
3. For each $p_h \in P$
      Calculate its Accuracy, Acc($p_h$), on labeled data
   Delete any cluster $p'$ whose Acc($p'$) < Avg_acc($P$)
4. Repeat till All centroids remain unchanged or the value of Eq. (3) decreases.
      4.1 Assign $x_i \in X_L \cup X_U$ to the closest cluster
      4.2 Update all centroids on $X_L \cup X_U$ using Eq. (1)
Output: the final cluster set $P$.
```

3. Hybrid resampling via semi-supervised hierarchical clustering

3.1. Motivation

Generally, existing clustering-based resampling methods separately divide majority or/and minority samples into several clusters, within which resampling is performed to achieve a class balance in the training set. Losing a global view of the data distribution may bring a negative impact on resampling. As shown in Fig. 1(b), undersampling may delete some informative majority samples near the cluster borderline.

On the other hand, oversampling may generate noises or outliers in overlapping or sparse areas. Moreover, the small disjunct is strongly underrepresented and likely to be treated as outliers. These problems will become even more severe when there is insufficient labeled data.

Moreover, the class information of labeled data is rarely utilized in existing clustering-based methods, so it is difficult to find an optimum partition for the resampling. For example, it remains a challenging issue to determine the cluster number and initial centroids for Kmeans, which is used in most clustering-based methods. With the support of information from labeled data, semi-supervised clustering is naturally suitable to capture the data distribution for the resampling and following classification.

Based on our previous work (Jiang et al., 2022), we propose a semi-supervised hierarchical Kmeans method to assist the hybrid resampling in CIL. The underlying assumption is that data in one "compact" cluster should share the same class label. With the supervision of labeled data, the proposed algorithm iteratively splits original classes into several clusters till all clusters are "compact" enough. As shown in Fig. 1(c), such clustering can efficiently capture the underlying data distribution for resampling. The cluster-splitting discloses inter-class and intra-class imbalance and identifies small disjuncts and overlapping areas. In the following subsection, we provide a detailed description of the proposed semi-supervised hierarchical Kmeans algorithm.

3.2. The semi-supervised hierarchical clustering algorithm (SSHC)

As a general framework, the CSSC (Jiang et al., 2022) iteratively refines the partition of traditional clustering algorithms via a cluster-splitting technique. Specifically, a cluster is split if it contains labeled data from different classes. In order to decrease the impact of noise, CSSC defines an objective function that consists of a traditional clustering loss and an item used to measure the compact degree. However, the class-imbalance issue is not taken into account by the CSSC framework.

```
Algorithm 2: The hybrid resampling via SSHC
  Input: $D_L, D_U$, the resampling ratio $\lambda$, and a threshold value, $min\_size$
  1. Perform Algorithm 1 on $D_L \cup D_U$ to get a data partition $P$.
  2. Undersampling majority class
      2.1 For each majority cluster $p_i^- \in P$ whose size $|p_i^-| < min\_size$
          Delete $p_i^-$ from $P$ and remove all labeled data in $p_i^-$ from $D_L$
      2.2 For each impure minority cluster $p_i^+$ in $P$
          Delete labeled data of majority class in $p_i^+$ from $D_L$
      2.3 Compute the weights of majority samples according to Eq. (5)
      2.4 Remove $n_U$ majority samples according to their weights
  3. Oversampling minority class
      3.1 Compute the confidence of unlabeled data in minority clusters according to Eq. (7)
      3.2 Add $n_O$ unlabeled data to $D_L$ according to their confidence
      3.3 If unlabeled data are deficient, then use SMOTE to compensate for the deficiency
  Output: the modified $D_L$
```

Some minority data may be assigned to majority clusters to fit the overall distribution of labeled data. In this paper, we present the SSHC algorithm which incorporates an item with respect to F-measure into the loss function to improve the performance on class-imbalanced datasets.

To begin with, we define the learning problem. Suppose X is an instance space defined on an unknown distribution that is approximated by a clustering model Θ, $Y = \{y_1, \dots, y_C\}$ is the corresponding class set. Let $D_L = \{X_L, Y_L\} = \{(x_i, y_i) | i = 1, 2, \dots, L\}$ represent the labeled data and $D_U = \{X_U\} = \{(x_j) | j = 1, 2, \dots, U\}$ represent the unlabeled data whose labels Y_U are unobserved. Let $P = \{p_1, \dots, p_K\}$ be the cluster set, and $\forall p_k \in P$ has a label $s_k \in Y$ which is determined by the dominant labeled data in p_k. Let $M = \{m_1, \dots, m_K\}$ denote the centroid matrix of P, where each m_k is the kth centroid, which is estimated as:

$$m_k = \frac{1}{|p_k|} \sum_{x_i \in p_k} x_i \tag{1}$$

where $|p_k|$ is the size of cluster p_k. Correspondingly, the original loss function of Kmeans is defined as:

$$SSE(X;\Theta) = \sum_{j=1}^{K} \sum_{x_i \in p_j} \left\| x_i - m_j \right\| \tag{2}$$

where $SSE(X;\Theta)$ denotes the Sum of Square Error. In a clustering model, it can be interpreted as the sum of the squared distance between the centroid and each cluster member. However, we argue that SSE is unsuitable for accessing a top-down hierarchical clustering since it may naturally decrease during the cluster-splitting process. Moreover, SSE will further decrease with the centroid updating. Therefore, we redefine the objective function of SSHC as:

$$J\left(X_L, X_U, Y_L, Y'_L;\Theta\right) = \mu \operatorname{Acc}\left(Y_L, Y'_L;\Theta\right) + (1-\mu) F1\left(Y_L, Y'_L;\Theta\right) \tag{3}$$

where Y'_L is the class label vector of X_L, which is predicted by the semi-supervised clustering model Θ. The first item Acc is the prediction accuracy of labeled data, and the second item $F1 \in [0,1]$ is a widely used measurement in CIL, which is estimated as:

$$F1\left(Y_L, Y'_L;\Theta\right) = \frac{2 \times TP \times TN}{TP(TP+FP) + TN(TP+FN)} \tag{4}$$

Where TP is the number of minority samples that are correctly classified, TN is the number of majority samples that are correctly classified, FP is the number of majority samples that are incorrectly classified, and FN is the number of minority samples that are incorrectly classified. In contrast with the first item, the second item focuses more on the performance on the minority class. The $\mu \in [0,1]$ is a weighting parameter to balance the impacts of two items in Eq. (3).

The motivation of the proposed objective function is to guide SSHC towards a solution that achieves an optimum imbalanced classification performance. It is quite different from traditional cluster algorithms because SSHC is designed to assist the CIL rather than purely disclosing the data distribution. The clustering model Θ is then estimated by maximizing the proposed objective function in Eq. (3), to seek an optimum data partition that not only reveals the data distribution but also respects the class information as much as possible. The corresponding solution is described in Algorithm 1.

In Step 1, the cluster model is initialized using D_L, and then a top-down splitting technique is proposed in Step 2 to find the optimal cluster number and initial centroids. The cluster-splitting helps to identify small disjuncts. To decrease the impact of noises or outliers on cluster-splitting, we delete the majority clusters whose accuracies are below the average accuracy of all clusters in Step 3. In Step 4, the centroids are updated on the whole dataset. In order to prevent the degradation of clustering performance, the objective function in Eq. (3) is used to supervise the cluster-splitting and the centroid-updating in Step 2 and Step 3, respectively.

3.3. *The hybrid resampling algorithm*

Based on the clustering result of SSHC, we present a hybrid resampling method in Algorithm 2 for class-imbalanced learning. It is noteworthy that some clusters may contain labeled data from different classes due to the supervision of the proposed objective function. Thus, each cluster's label is determined by its dominant labeled data. Subsequently, undersampling and oversampling are respectively performed to achieve a given degree of class balance. In Algorithm 2, we define a resampling ratio, λ, to compute undersampling number, n_U, and oversampling number, n_O, in Eq. (6) and (8), respectively.

3.3.1. *Undersampling majority class*

In step 2, we first delete small-size majority clusters that are likely to be outliers. Here, the threshold value min_size is set as 10 percent of the average size of majority clusters. Next, we delete majority samples in minority clusters which are considered noises. This step also contributes to decreasing the impact of class overlapping problems. Finally, the remained majority samples are undersampled according to the undersampling weight. For a majority sample x_i, its undersampling weight $w_i \in (0,1)$ is defined as:

$$w_i = \frac{1}{1 + \exp\left(d_i^- - d_i^+\right)} \tag{5}$$

Where d_i^+ is the distance of x_i to its own cluster centroid and d_i^- is the distance of x_i to the closest minority cluster centroid. In this way, a majority sample that is far away from its centroid but near the minority cluster is imposed a large undersampling weight. In addition to avoiding deleting informative samples, these steps contribute to biasing the decision boundary towards the majority class. To balance the training set, the number of majority samples to be deleted is estimated as:

$$n_U = \lambda * \left(N^- - N^+\right) \tag{6}$$

where λ is the resampling ratio, N^+ and N^- denote the number of minority and majority samples, respectively.

3.3.2. Oversampling minority class

Traditional oversampling methods resample from original labeled data or generate new synthetic data. Although these methods enlarge the training set, they do not help discover more about the actual data distribution. In step 3, we present a new oversampling method which selects unlabeled data from minority clusters as samples for the minority class. The underlying assumption is that data in one cluster should share the same label. Furthermore, we define the confidence that an unlabeled data x in cluster p_j is labeled as y_i:

$$C\left(y_i \mid x;\Theta\right) = \frac{n^+}{\mathrm{n}}\left(1 - \frac{\left\|x - m_j\right\|}{\sum_{k=1}^{|P|}\left\|x - m_k\right\|}\right) \tag{7}$$

where m_j is the centroid of cluster p_j, n and n^+ denote the number of samples and minority samples in p_j, respectively. Essentially, the label confidence of unlabeled data is negatively correlated with the distance to its centroid and positively correlated with the distance to other centroids. Oversampling from unlabeled data provides more details of the data distribution which cannot be described by original training data. Moreover, it alleviates the impact of a small training sample size. The number of unlabeled data to be added as minority samples is estimated as:

$$n_O = (1 - \lambda) * \left(N^- - N^+\right) \tag{8}$$

It should be noted that the semi-supervised clustering also suffers from the imbalanced distribution, i.e., the data of minority class are likely to be assigned to majority clusters. When the number of unlabeled data in minority clusters is less than n_O, the standard SMOTE is employed in step 3.3 to compensate for the deficiency.

3.4. Computational complexity

Let l, n, and m denote the numbers of labeled data, unlabeled data, and the features, respectively. In Algorithm 1, the cluster number increases with the cluster-splitting in Step 2, and remains unchanged in Step 4. Let K_1 is the average cluster number in Step 2, and K_2 is the cluster number in Step 3. We have $|Y| \le K_1 \le K_2$. The computational complexity of SSHC is $O(m(l+u)(t_1K_1|Y| + t_2K_2))$, where t_1 and t_2 is the iteration number in Step 2 and Step 4, respectively. Let t_{max} denote $max\{t_1, t_2\}$, we have $m(l+u)(t_1K_1|Y| + t_2K_2) < m(l+u)t_{max}K_2(|Y|+1)$. The computational complexity of SSHC can be roughly estimated as $O(m(l+u)t_{max}K_2|Y|)$. By contrast, existing clustering-based resampling methods only cluster labeled data. Taking the widely used Kmeans for an example, the computational complexity of clustering is $O(mltK)$, where t and K denote the cluster number and the iteration number of centroid-updating, respectively. Although SSHC runs on a larger number of data, its iteration number, t_1 and t_2, we argue, is far less than t, due to the supervision of the proposed objective function. In general, SSHC enjoys almost the same scale of computational complexity as Kmeans-style methods used in most clustering-based resampling methods.

In Algorithm 2, we search each cluster and remove majority samples that might be noises or outliers in Step 2.1 and 2.2. The computational complexity is $O(ml^*K_2)$, where l^* is the maximum size of each cluster. In Step 2.3, we compute the weights of majority samples according to Eq. (5), whose computational complexity is $O(mlK_2)$. The computational complexity of removing n_U majority samples in Step 2.4 is $O(mln_U)$. In Step 3.1, the computational complexity of computing the weights of minority samples is $O(muK_2)$, and the computational complexity of selecting nO unlabeled data in Step 3.2 is $O(mun_O)$. Moreover, the SMOTE is used to compensate for the deficiency of unlabeled data in Step 3.3. Its computational complexity may be ignored because the SMOTE is performed within clusters and the number of samples to be generated is usually tiny.

In summary, the total computational complexity of Algorithm 2 can be estimated as $O(ml^*K_2 + mlK_2 + mln_U + muK_2 + mun_O)$. Note that $l^*, n_U, n_O < l$, so we have $ml^*K_2 + mlK_2 + mln_U + muK_2 + mun_O < 2mlK_2 + ml^2 + muK_2 + mul = mK_2(2l+u) + m(l+u)$. Generally, the computational complexity of Algorithm 2 can be simplified as $O(mK_2(l+u))$. On the other hand, the computational complexities of existing resampling methods usually have the scale of $O(ml^2)$ in the worst case. Although the usage of unlabeled data increases the data scale of SSHR, the clustering result of SSHC significantly decreases the time-consuming of SSHR. In contrast with traditional resampling methods, we only need to compute the data's distances to cluster centroids rather than the distances among samples when performing the resampling.

Table 1
Confusion matrix.

	Predict positive class	Predict negative class
True positive class	TP	FN
True negative class	FP	TN

4. Experiments

To evaluate the performance of the proposed method, we present the experimental comparison with nine state-of-the-art resampling methods. The Synthetic Minority Over-sampling Technique (SMOTE) (Chawla et al., 2002), adaptive synthetic (ADASYN) sampling method (He et al., 2008), and SMOTE-Rough Set Theory (sRST) method (Ramentol et al., 2012), and A-SUWO (Nekooeimehr & Lai-Yuen, 2016), Evolutionary Under-Sampling (EUS) method (García & Herrera, 2009) and Trainable Undersampling (TU) (Peng et al., 2019) are under-sampling methods. Fuzzy rule base + Cross-generational elitist selection, Heterogeneous recombination and Cataclysmic mutation (FRB +CHC) (Wong, Leung, & Ling, 2018) and Clustering and Density-Based Hybrid (CDBH) method (Mirzaei et al., 2021) are hybrid resampling methods, and Self-paced ensemble method (SPE) (Liu et al., 2020)is an undersampling-based ensemble method.

4.1. Evaluation measures

We employ F-Measure and AUC as the evaluation measures in the experiment. Both evaluation measures are dependent on the confusion matrix, as illustrated in Table 1. The TP (True Positive) is the number of positive class (minority class) samples that are correctly classified, and TN (True Negative) is the number of negative class (majority class) samples that are correctly classified. The FP (False Positive) is the number of negative samples which are incorrectly classified as positive, and FN (False Negative) is the number of positive samples that are incorrectly classified as negative.

Six performance metrics for classification tasks are then defined as follows:

Recall =TP / (TP + FN)
Precision =TP / (TP + FP)
Specificity=FP / (TN + FP)
Sensitivity =FN / (TP + FN)

$$\text{F-Measure} = \frac{(1+\beta) * \text{ Recall } * \text{ Precision}}{\beta^2 * \text{ Recall } + \text{ Precision}} \tag{9}$$

where β is a coefficient to describe the relative importance of precision and recall rates, which is usually set to 1.

$$\text{AUC} = (1 + \text{ Sensitivity - Specificity })/2 \tag{10}$$

The F-measure represents a harmonic mean between Recall and Precision. The AUC is referred to as the area under ROC graph and

Table 2
Descriptions of datasets.

Datasets	#Samp	#Attr	Min,Maj.(%)	I.R.
Abalone19	4174	8	(0.7,99.23)	129.44
Abalone918	731	8	(5.65,94.25)	16.4
Cleveland0vs4	177	13	(7.34,92.66)	12.62
Ecoli0137vs26	281	7	(2.49,97.51)	39.14
Ecoli0146vs5	280	6	(7.14,92.86)	13
Ecoli0147vs2356	336	7	(8.63,91.37)	10.59
Ecoli0147vs56	332	6	(7.53,92.47)	12.28
Ecoli01vs235	244	7	(9.83,90.17)	9.17
Ecoli01vs5	240	6	(8.33,91.67)	11
Ecoli0234vs5	202	7	(9.9,90.1)	9.1
Ecoli0267vs35	224	7	(9.82,90.18)	9.18
Ecoli0346vs5	205	7	(9.76,90.24)	9.25
Ecoli0347vs56	257	7	(9.73,90.27)	9.28
Ecoli034vs5	200	7	(10,90)	9
Ecoli046vs5	203	6	(9.85,90.15)	9.15
Ecoli067vs35	222	7	(9.91,90.09)	9.09
Ecoli067vs5	220	6	(9.09,90.91)	10
Ecoli4	336	7	(5.95,94.05)	15.8
Glass0146vs2	205	9	(8.29,91.71)	11.06
Glass015vs2	172	9	(9.88,90.12)	9.12
Glass016vs2	192	9	(8.85,91.15)	10.29
Glass016vs5	184	9	(4.89,95.11)	19.44
Glass04vs5	92	9	(9.78,90.22)	9.22
Glass06vs5	108	9	(8.33,91.67)	11
Glass2	214	9	(7.94,92.06)	11.59
Glass4	214	9	(6.07,93.93)	15.47
Glass5	214	9	(4.2,95.8)	22.78
Leddigit02456789vs1	443	7	(8.35,91.65)	10.97
Page_blocks13vs4	472	10	(5.93,94.07)	15.86
Shuttlec0vsc4	1829	9	(6.72,93.28)	13.87
Shuttlec2vsc4	129	9	(4.65,95.35)	20.5
Vowel0	988	13	(9.01,90.99)	9.98
Yeast0256vs3789	1004	8	(9.86,90.14)	9.14
Yeast02579vs368	1004	8	(9.86,90.14)	9.14
Yeast0359vs78	506	8	(9.88,90.12)	9.12
Yeast05679vs4	528	8	(9.66,90.34)	9.35
Yeast1289vs7	947	8	(3.16,96.84)	30.57
Yeast1458vs7	693	8	(4.33,95.67)	22.1
Yeast1vs7	459	7	(6.53,93.47)	14.3
Yeast2vs4	514	8	(9.92,90.08)	9.08
Yeast2vs8	482	8	(4.15,95.85)	23.1
Yeast4	1484	8	(3.43,96.57)	28.1
Yeast5	1484	8	(2.96,97.04)	32.73
Yeast6	1484	8	(2.36,97.64)	41.4

is not sensitive to the distribution of two classes. In principle, the AUC criterion evaluates the overall accuracy of the classifier over both the majority and minority classes. In contrast, the F-measure criterion only evaluates the accuracy of the classifier over the minority class.

4.2. Setup

In the experimental part, we use the same datasets and experimental setup as CDBH (Mirzaei et al., 2021) for convenient and fair comparison.

Datasets:

The proposed method is evaluated on 44 imbalanced datasets from the KEEL repository,[1] whose detailed information is shown in Table 2.

Parameter setup:

For these resampling methods, we adopted the Support Vector Machine (SVM) as the base classifier with a radial basis function (RBF) as its kernel.

$$RBF = \exp\left(-\frac{1}{\sigma}\left\|x_i - x\right\|^2\right) \tag{11}$$

[1] http://www.keel.es/

Following the setting of CDBH, the parameter σ and the tradeoff between training error and margin of SVM are set to 0.01 and 100, respectively. For SMOTE, the number of nearest minority neighbors (NN) found for each minority instance to generate synthetic samples is set as 5. Moreover, we use the original settings of all compared algorithms for the experimental validation. Our method has three hyperparameters, μ, min_size and λ, which are empirically set as 0.5,0.1 and 0.3, respectively.

4.3. Experimental results

Following the experimental setting of CDBH, five-fold stratified cross-validation was used, and each experiment was repeated five times to report their averaged values to avoid randomness influences on the results. Tables 3 and 4 show the performances of different algorithms on 44 datasets, with respect to F-Measure and AUC, respectively. Moreover, the performance of standard SVM is listed as a baseline for the experimental comparison. The detailed analyses are provided in the following.

4.3.1. The effect of the SSHC algorithm

To deal with the class-imbalanced problem, we use the proposed semi-supervised clustering algorithm, SSHC, to decompose original classes into multiple clusters and then select pseudo-labeled data for minority class. In order to illustrate the effect of SSHC on resampling, we list the results of SSHC in Table 7, where C+, C-, and P_N denote the number of minority clusters, majority clusters and selected unlabeled data. The Acc denotes the accuracy of pseudo-labeled data.

It is observed that a various number of clusters (subclasses) are identified by SSHC on different datasets. The majority class is divided into more clusters than the minority class. Obviously, the decomposition of majority class helps to identify inter-class imbalance, while the decomposition of minority class helps to identify the intra-class imbalance. The class overlapping problem can also be addressed by splitting involved classes into several small-size clusters. As a result, we observed in Tables 3 and 4 that our method significantly outperforms SVM, especially on those well-split datasets. These results demonstrate that our resampling method benefits from the cluster-splitting performed by SSHC. However, nearly no sub-clusters are identified on $Yeast2vs8$, $Shuttlec2vsc4$ and $Shuttlec0vsc4$. The main reason may lie in their compact and well-separated classes.

The resampling is then performed on the result of SSHC. Majority samples are under-sampled according to the distances to their own cluster centroids and the adjacency to minority cluster centroids. In addition, the SSHC takes unlabeled data near the centroids of minority clusters as pseudo-labeled data of the minority class. Compared with traditional over-sampling methods, our oversampling method can disclose more about the underlying distribution of the minority class. Moreover, an additional benefit is that taking advantage of unlabeled data alleviates the deficiency of labeled data in many real-world tasks. It is another factor why our resampling method significantly improves the CIL performance. As shown in Table 7, a various number of pseudo-labeled data are selected for different datasets. Correspondingly, the performance improves with the addition of pseudo-labeled data. Although in some datasets, such as $Abalone19$, $Glass016vs5$, and *Yeast1458vs7*, the accuracies of pseudo-labeled data are relatively low, the proposed method achieves significant superiority over the original classifier. These results demonstrate that our method is robust to the noise among pseudo-labeled data. In some ways, we argue that moderate noises among minority samples might facilitate identifying the minority class. Compared with the original classifier, the performance of our method increases by 39.3% and 24.8% for F-measure and AUC, respectively. It demonstrates the effectiveness of using SSHC to improve resampling.

Table 3
The result measured by F-measure.

Datasets	Original	SMOTE	ADASYN	sRST	A-SUWO	EUS	FRB+CHC	CDBH	Spe	Tu	SSHR
Abalone19	0.000	0.041	0.041	0.049	0.000	0.000	0.048	0.017	0.026	0.000	**0.053**
Abalone918	0.044	0.452	0.417	0.457	0.446	0.264	**0.573**	0.431	0.272	0.568	0.480
Cleveland0vs4	0.000	0.154	0.156	0.160	0.471	0.262	0.169	0.179	0.078	0.000	**0.555**
Ecoli0137vs26	0.000	0.398	0.240	0.364	0.450	0.316	0.347	0.552	0.028	0.000	**0.700**
Ecoli0146vs5	0.000	0.428	0.192	0.442	0.695	0.699	0.746	**0.840**	0.219	0.750	0.796
Ecoli0147vs2356	0.000	0.423	0.301	0.444	0.286	0.223	0.404	0.711	0.228	0.721	**0.742**
Ecoli0147vs56	0.000	0.576	0.415	0.510	0.786	0.615	0.676	0.754	0.248	0.809	**0.852**
Ecoli01vs235	0.000	0.433	0.165	0.426	0.767	0.569	0.422	**0.784**	0.265	0.667	0.734
Ecoli01vs5	0.000	0.414	0.259	0.495	0.763	0.414	0.684	**0.854**	0.264	0.750	0.795
Ecoli0234vs5	0.000	0.518	0.292	0.473	0.801	0.668	0.614	0.809	0.313	0.733	**0.833**
Ecoli0267vs35	0.000	0.316	0.227	0.325	0.752	0.404	0.459	0.627	0.293	0.750	**0.777**
Ecoli0346vs5	0.000	0.545	0.364	0.564	0.821	0.738	0.677	0.837	0.329	0.817	**0.860**
Ecoli0347vs56	0.000	0.574	0.474	0.510	0.782	0.667	0.518	0.736	0.305	0.769	**0.848**
Ecoli034vs5	0.000	0.563	0.267	0.501	0.771	0.659	0.583	0.814	0.334	0.749	**0.842**
Ecoli046vs5	0.000	0.390	0.200	0.421	0.791	0.679	0.523	0.835	0.315	**0.875**	0.862
Ecoli067vs35	0.000	0.454	0.398	0.445	0.758	0.476	0.431	0.639	0.293	0.765	**0.780**
Ecoli067vs5	0.000	0.326	0.297	0.323	0.813	0.685	0.617	0.720	0.286	0.571	**0.831**
Ecoli4	0.750	0.635	0.508	0.649	0.770	0.737	0.736	0.819	0.853	**0.881**	0.821
Glass0146vs2	0.000	0.246	0.251	0.281	0.296	0.175	0.260	**0.423**	0.216	0.000	0.309
Glass015vs2	0.000	0.309	0.318	0.342	0.177	0.102	0.205	**0.379**	0.210	0.040	0.276
Glass016vs2	0.000	0.320	0.320	0.296	0.290	0.140	0.200	**0.355**	0.226	0.044	0.303
Glass016vs5	0.665	0.567	0.659	0.655	0.810	0.469	0.769	0.603	0.541	0.740	**0.863**
Glass04vs5	**1.000**	0.879	0.868	0.921	0.760	0.785	0.963	0.570	**1.000**	**1.000**	**1.000**
Glass06vs5	**1.000**	0.906	0.966	0.908	0.733	0.865	0.978	0.733	0.737	0.700	**1.000**
Glass2	0.000	0.248	0.236	0.299	0.273	0.113	0.202	**0.405**	0.218	0.000	0.262
Glass4	**0.856**	0.663	0.657	0.646	0.674	0.716	0.727	0.554	0.648	0.791	0.760
Glass5	0.700	0.594	0.455	0.484	0.810	0.354	0.753	0.607	0.402	0.800	**0.888**
Leddigit02456789vs1	**0.775**	0.571	0.620	0.516	0.746	0.569	0.675	0.725	0.726	0.750	0.748
Page_blocks13vs4	0.227	0.203	0.191	0.189	0.733	0.083	0.182	**0.948**	0.155	0.333	0.672
Shuttlec0vsc4	0.949	0.974	0.894	0.982	**0.996**	0.971	0.796	0.977	0.957	0.939	**0.996**
Shuttlec2vsc4	0.400	0.715	0.715	0.729	0.400	0.159	0.613	**0.933**	0.108	0.333	0.853
Vowel0	**1.000**	0.994	0.980	0.982	0.962	0.940	0.906	0.992	**1.000**	**1.000**	0.853
Yeast0256vs3789	0.178	0.528	0.439	0.533	0.442	**0.603**	0.590	0.501	0.527	0.590	0.551
Yeast02579vs368	0.815	0.720	0.521	0.720	0.798	0.749	0.775	0.749	0.786	**0.827**	0.755
Yeast0359vs78	0.348	0.354	0.297	0.353	0.382	0.348	0.347	0.385	0.270	0.334	**0.403**
Yeast05679vs4	0.000	0.433	0.427	0.425	**0.577**	0.500	0.479	0.467	0.458	0.521	0.472
Yeast1289vs7	0.000	0.140	0.136	0.131	**0.229**	0.000	0.197	0.204	0.111	0.000	0.164
Yeast1458vs7	0.000	0.132	0.126	0.134	0.094	0.000	0.156	**0.158**	0.108	0.106	0.141
Yeast1vs7	0.000	0.293	0.287	0.274	**0.350**	0.000	0.316	0.335	0.246	0.279	0.320
Yeast2vs4	0.638	0.682	0.545	0.679	**0.800**	0.742	0.702	0.675	0.752	0.783	0.736
Yeast2vs8	**0.697**	0.597	0.208	0.598	0.686	**0.697**	**0.697**	0.343	0.651	0.651	0.666
Yeast4	0.000	0.270	0.246	0.271	**0.378**	0.031	0.353	0.285	0.313	0.017	0.321
Yeast5	0.000	0.484	0.461	0.475	0.569	0.580	0.448	0.628	**0.635**	0.578	0.527
Yeast6	0.000	0.270	0.201	0.267	0.385	0.000	0.329	0.413	0.388	**0.525**	0.350
Mean	0.251	0.471	0.392	0.469	0.593	0.449	0.519	0.598	0.394	0.542	**0.644**

4.3.2. The comparison with other resampling algorithms

Tables 3 and 4 show that our method outperforms its competitors on average results. Besides, our method performs best on 18 out of 44 and 16 out of 44 datasets, with respect to F-Measure and AUC, respectively. Compared to the original SVM, all resampling methods achieve noticeable improvements, which show the efficiency of these resampling methods in handling class-imbalanced problems. We also noticed that the increase of F-Measure is more significant than that of AUC. This result demonstrates that these resampling methods effectively identify minority samples, considering that F-Measure focuses more on the minority class than AUC. The SSHR performs better than the second-best method by 4.6% and 3.8%, in terms of F-Measure and AUC, respectively. We argue that the success of SSHR is largely due to SSHC which can better disclose the data distribution. Based on the result of SSHC, the hybrid resampling contributes to addressing not only inter-class imbalance but also intra-class imbalance. In addition, the incorporation of pseudo-labeled data improves the performance of SSHR further.

Furthermore, we perform a non-parametric statistical test called the Friedman test (García, Fernández, Luengo, & Herrera, 2010)to assess whether there are significant performance differences among these focused methods. First, we provide the average ranks of different algorithms in Table 5. The lower average ranking means better performance of the desired method than other methods, so the winner method has the lowest average ranking. We observed that our method has the best ranking for both F-measure and AUC and the superiority is obvious. Moreover, other hybrid resampling methods, FRB+CHC and CDBH show good performance on class-imbalanced datasets. It demonstrates the benefit of leveraging the strength of both undersampling and oversampling. It is worth noting that A-SUWO ranks third in nine CIL methods. The main reason, we guess, is that A-SUWO uses a semi-unsupervised clustering to utilize the class information to guide the oversampling.

In the next step, a post hoc procedure, Holm's test, is conducted to assess the differences among methods in Table 6 with a significance level ($\alpha = 0.05$). If the p-value is lower than the Holm value, the null hypothesis will be rejected, i.e., the difference is significant. The results in Table 6 show the significant superiority of SSHR over other algorithms except for A-SUWO. Therefore, SSHR not only attains the best F-measure ranking but also significantly outperforms eight state-of-the-art resampling methods. In conclusion, the experimental results demonstrate the effectiveness of utilizing SSHR to improve the performance on class-imbalanced datasets.

4.3.3. The impact of the hyperparameters

There are three hyperparameters required in our method. The first one, $\mu \in [0,1]$,is used in Algorithm 1 as a weighting parameter to balance the impacts of two items, Acc and F1, in the objective function. A larger μ means giving larger weight to Acc. The second one, min_size,

Table 4
The result measured by AUC.

Datasets	Original	SMOTE	ADASYN	sRST	A-SUWO	EUS	FRB+CHC	CDBH	Spe	Tu	SSHR
Abalone19	0.500	0.718	0.717	**0.772**	0.500	0.500	0.702	0.508	0.679	0.500	0.757
Abalone918	0.513	**0.896**	0.886	0.892	0.698	0.579	0.860	0.774	0.743	0.700	0.843
Cleveland0vs4	0.497	0.562	0.558	0.542	0.721	0.599	0.586	0.562	0.359	0.500	**0.840**
Ecoli0137vs26	0.500	0.712	0.593	0.691	0.741	0.643	0.666	0.839	0.306	0.500	**0.895**
Ecoli0146vs5	0.498	0.644	0.565	0.656	0.840	0.773	0.837	**0.919**	0.725	0.865	0.901
Ecoli0147vs2356	0.498	0.651	0.615	0.663	0.583	0.650	0.805	0.857	0.667	0.824	**0.868**
Ecoli0147vs56	0.500	0.716	0.635	0.705	0.909	0.772	0.858	0.890	0.749	0.893	**0.918**
Ecoli01vs235	0.496	0.661	0.538	0.662	0.876	0.742	0.787	**0.906**	0.694	0.750	0.840
Ecoli01vs5	0.498	0.660	0.586	0.688	0.866	0.709	0.816	**0.936**	0.741	0.864	0.890
Ecoli0234vs5	0.497	0.698	0.614	0.682	**0.908**	0.801	0.829	0.899	0.750	0.859	0.906
Ecoli0267vs35	0.500	0.607	0.583	0.611	**0.860**	0.704	0.818	0.850	0.735	0.800	0.858
Ecoli0346vs5	0.497	0.697	0.612	0.713	0.892	0.788	0.846	**0.925**	0.776	0.890	0.919
Ecoli0347vs56	0.500	0.757	0.703	0.729	0.905	0.807	0.789	0.884	0.750	0.865	**0.923**
Ecoli034vs5	0.497	0.707	0.589	0.680	0.864	0.811	0.822	**0.908**	0.778	0.868	0.907
Ecoli046vs5	0.497	0.650	0.561	0.670	0.892	0.746	0.788	0.912	0.755	**0.974**	0.920
Ecoli067vs35	0.500	0.686	0.663	0.670	0.855	0.668	0.786	0.852	0.730	0.853	**0.867**
Ecoli067vs5	0.500	0.610	0.610	0.611	0.893	0.800	0.813	0.870	0.743	0.737	**0.898**
Ecoli4	0.800	0.910	0.915	0.943	0.914	**0.953**	0.937	0.913	0.920	0.916	0.940
Glass0146vs2	0.500	0.682	0.685	0.714	0.667	0.552	0.634	**0.749**	0.664	0.500	0.730
Glass015vs2	0.500	0.715	0.735	**0.750**	0.637	0.491	0.553	0.675	0.586	0.519	0.672
Glass016vs2	0.500	**0.753**	**0.753**	0.732	0.603	0.573	0.611	0.672	0.663	0.528	0.712
Glass016vs5	0.844	0.886	0.919	0.922	0.980	0.807	0.919	0.841	0.949	0.846	**0.981**
Glass04vs5	**1.000**	0.975	0.975	0.983	0.919	0.957	0.973	0.779	**1.000**	**1.000**	**1.000**
Glass06vs5	**1.000**	0.977	0.995	0.944	0.850	0.940	0.984	0.849	0.950	0.850	**1.000**
Glass2	0.500	0.713	0.698	**0.761**	0.671	0.525	0.608	0.724	0.691	0.500	0.681
Glass4	0.909	0.915	0.918	0.916	0.939	0.925	0.923	0.880	0.929	0.906	**0.940**
Glass5	0.845	0.876	0.826	0.852	0.983	0.877	0.897	0.796	0.934	**0.988**	0.956
Leddigit02456789vs1	0.879	0.882	0.887	0.865	0.863	**0.906**	0.884	0.863	0.899	0.884	0.901
Page_blocks13vs4	0.570	0.753	0.732	0.730	0.959	0.561	0.714	**0.989**	0.658	0.600	0.798
Shuttlec0vsc4	0.952	0.975	0.987	0.985	0.996	0.972	0.981	**0.998**	0.960	0.943	0.996
Shuttlec2vsc4	0.700	0.955	0.955	0.959	0.700	0.696	0.949	**0.987**	0.587	0.650	0.936
Vowel0	**1.000**	0.999	0.998	0.998	0.991	0.993	0.989	0.998	**1.000**	**1.000**	0.982
Yeast0256vs3789	0.549	0.796	0.773	0.799	0.729	**0.806**	0.769	0.764	0.804	0.782	0.804
Yeast02579vs368	0.870	0.906	0.861	0.907	0.896	**0.914**	0.913	0.895	0.907	0.885	0.908
Yeast0359vs78	0.607	0.734	0.694	0.733	0.677	0.607	0.606	0.708	0.673	0.607	**0.744**
Yeast05679vs4	0.500	0.787	0.790	0.780	0.784	0.786	0.790	0.757	**0.792**	0.736	0.791
Yeast1289vs7	0.500	0.714	0.715	0.697	0.600	0.500	0.699	0.683	0.694	0.500	**0.743**
Yeast1458vs7	0.500	0.643	0.637	0.654	0.571	0.500	0.596	0.601	0.612	0.530	**0.664**
Yeast1vs7	0.500	0.758	**0.774**	0.750	0.690	0.500	0.693	0.717	0.727	0.638	0.769
Yeast2vs4	0.736	**0.892**	0.879	0.889	0.891	0.880	0.876	0.869	0.892	0.864	0.890
Yeast2vs8	0.774	0.763	0.724	**0.777**	0.774	0.774	0.774	0.703	0.774	0.774	0.774
Yeast4	0.500	0.816	0.810	0.812	0.784	0.509	0.799	0.745	**0.818**	0.505	0.817
Yeast5	0.500	0.967	0.964	0.966	0.952	0.798	0.962	0.937	0.930	0.859	**0.972**
Yeast6	0.500	0.874	0.860	0.874	0.832	0.500	0.888	0.854	0.884	0.845	**0.890**
Mean	0.614	0.778	0.752	0.780	0.810	0.725	0.803	0.824	0.763	0.759	**0.862**

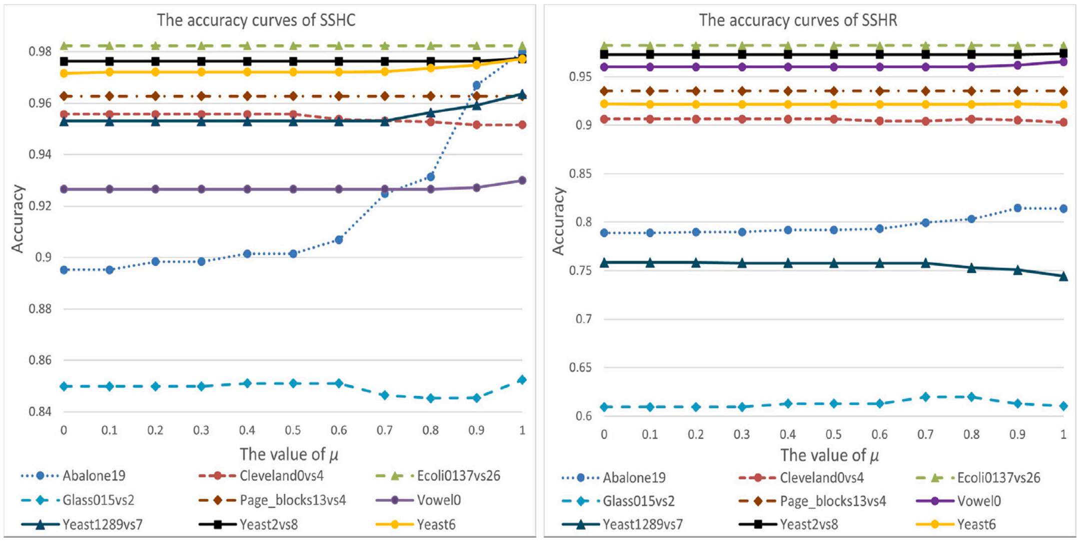


Fig. 2. The impact of the hyperparameter μ on SSHC and SSHR.

Table 5
Average ranking of Friedman test.

Preprocessing methods	F-measure	AUC	Average
Original	8.64	9.70	9.17
SMOTE	6.48	5.68	6.08
ADASYN	8.12	6.62	7.37
sRST	6.47	5.42	5.95
A-SUWO	4.14	5.36	4.75
EUS	7.12	7.26	7.19
FRB+CHC	5.53	5.57	5.55
CDBH	4.07	5.27	4.67
Spe	7.53	5.71	6.62
Tu	5.17	6.90	6.04
SSHR	2.72	2.52	2.62

Table 6
Post Hoc comparison for $\alpha = 0.05$,used for the criterion F-measure.

Preprocessing methods	p values	Holm
CDBH	0.04871418	0.05
A-SUWO	0.04455667	0.025
Tu	0.00012442	0.0161
FRB+CHC	0.00000005	0.0125
sRST	0.00000000	0.01
SMOTE	0.00000000	0.0083
EUS	0.00000000	0.00714
Spe	0.00000000	0.00625
ADASYN	0.00000000	0.00556
Original	0.00000000	0.005

Table 7
The effect of the SSHR algorithm.

Datasets	C+	C-	P_N	Acc.
Abalone19	1.0	3.2	45.2	0.013
Abalone918	1.2	3.2	6.2	0.226
Cleveland0vs4	1.4	2.0	3.6	0.611
Ecoli0137vs26	1.0	1.4	1.6	0.625
Ecoli0146vs5	1.0	2.2	3.8	0.842
Ecoli0147vs2356	1.2	2.2	4.6	0.783
Ecoli0147vs56	1.0	2.0	4.2	0.810
Ecoli01vs235	1.0	2.0	4.0	0.800
Ecoli01vs5	1.0	2.0	4.0	0.750
Ecoli0234vs5	1.0	1.8	3.6	0.889
Ecoli0267vs35	1.2	2.2	3.8	0.842
Ecoli0346vs5	1.0	2.0	3.8	0.895
Ecoli0347vs56	1.0	2.0	4.6	0.870
Ecoli034vs5	1.0	2.0	3.6	0.889
Ecoli046vs5	1.0	2.0	3.8	0.895
Ecoli067vs35	1.6	2.6	4.0	0.800
Ecoli067vs5	1.0	2.0	4.0	0.800
Ecoli4	1.0	2.6	4.0	0.850
Glass0146vs2	1.6	4.4	3.8	0.211
Glass015vs2	1.6	3.8	1.4	0.143
Glass016vs2	3.6	7.4	3.8	0.158
Glass016vs5	1.0	2.4	1.2	0.167
Glass04vs5	1.2	1.8	1.0	0.600
Glass06vs5	1.0	2.0	1.0	0.800
Glass2	1.4	3.8	3.6	0.278
Glass4	1.8	3.2	3.4	0.471
Glass5	1.0	2.2	0.8	0.500
Leddigit02456789vs1	1.2	3.0	8.4	0.667
Page_blocks13vs4	1.0	2.0	2.8	0.857
Shuttlec0vsc4	1.0	1.0	24.4	1.000
Shuttlec2vsc4	1.0	1.0	1.2	0.833
Vowel0	1.0	2.0	19.8	0.576
Yeast0256vs3789	2.8	3.2	14.8	0.622
Yeast02579vs368	1.0	2.0	16.2	0.877
Yeast0359vs78	1.8	3.4	5.6	0.500
Yeast05679vs4	1.0	2.0	9.6	0.604
Yeast1289vs7	1.4	4.0	2.6	0.231
Yeast1458vs7	1.0	3.0	8.0	0.075
Yeast1vs7	1.8	3.8	5.2	0.462
Yeast2vs4	1.4	2.2	8.2	0.878
Yeast2vs8	1.0	1.0	2.4	0.917
Yeast4	1.0	2.4	13.6	0.324
Yeast5	1.0	2.0	10.8	0.593
Yeast6	1.2	2.8	10.2	0.412

is used in Algorithm 2 as a threshold value to delete those small-size majority clusters. Our purpose is to remove majority samples that are likely to be outliers. The value of min_size is set as:

$$min\_size = \delta * \text{Avg\_N} \tag{12}$$

where δ is a weight parameter, and Avg_N denotes the average size of majority clusters. The third hyperparameter $\lambda \in [0, 1]$ is used to determine the undersampling and oversampling ratios in Eqs. (6) and (8), respectively. A larger λ means increasing the undersampling ratio while decreasing the oversampling ratio. In the experimental results, μ, δ and λ is empirically set as 0.5, 0.1 and 0.3, respectively.

In the following, we explore the impact of three parameters on the effectiveness of our method on nine representative datasets. Firstly, we fix δ and λ to 0.1 and 0.3 respectively and vary μ from 0 to 1 to observe the impact of μ on the proposed method. Fig. 2(a) and (b) show the Accuracy curves for SSHC and SSHR, respectively. We observed that the accuracy curves for SSHC and SSHR show a general stable uptrend on $Abalone19$, and $Glass015vs2$. On $Yeast1289vs7$, the accuracy curve for SSHC goes up at the end while accuracy curve for SSHR ends with a decline. Generally, the variations of SSHC curves are larger than that of SSHR curves. The accuracy curves on the other six datasets remain roughly unchanged with the change of μ. In the objective function of SSHC, Acc and F1 focus on the overall performance and the performance of minority class, respectively. Both items have the same tends in many cases, which is why six datasets remain roughly unchanged with the change of μ. In other datasets, giving a larger weight to Acc naturally improves the performance measured by accuracy. Moreover, the variations of SSHC curves are generally larger than that of SSHR curves. The main reason is that the parameter μ has a direct impact on SSHC but an indirect impact on SSHR.

Secondly, we fix μ and λ to 0.5 and 0.3, respectively and vary δ from 0 to 0.5 to observe the performance changes of SSHR. Fig. 3(a) and (b) show the performance curves for F-measure and AUC, respectively.

Generally, the performance variances for different δ are relatively stable, especially on $page\_blocks13vs4$, $Yeast2vs8$ and $Yeast1289vs7$. As shown in Fig. 3, when δ varies from 0 to 0.5, most performance curves go up slightly at first, which shows the benefit of deleting possible outliers of majority class. After a relatively flat stage, most curves may end with a decline. These results show that the deletion of larger clusters has the risk of losing informative samples. Overall, the best performance in Fig. 3 is usually achieved when δ is between 0.1 and 0.3. The possible reason might be that there is an optimum balance between two factors when δ falls in this range. However, we noticed that the F-measure curve on $Page\_blocks13vs4$ declines sharply when δ varies from 0.2 to 0.4. Moreover, the AUC curve on $Glass015vs2$ declines when δ varies from 0.3 to 0.5. The possible reason is that many small-size majority clusters exist in these datasets. As a result, too many majority samples are deleted in this scenario.

Finally, we fix μ and δ to 0.5 and 0.1, respectively, to study the impact of λ. Fig. 4 shows the performance curves when λ varies from 0 to 0.9. Compared with the first hyperparameter δ, the resampling ratio λ has clearly larger impact on the performance. In Fig. 4(a), the F-measure curves on $Vowel0$ and $Cleveland0vs4$ remain a roughly stable tendency. The F-measure curves on $Glass015vs2$, $Abalone19$ and $Yeast1289vs7$ firstly grow up and then gradually decrease. By contrast, the F-measure curve on $Yeast6$ remains a steadily rising tendency. Notably, the F-measure curve on $Page\_blocks13vs4$ has a sharp increase, followed by a relatively flat stage. On the other hand, the AUC curves in Fig. 4(b) mostly remain a roughly stable tendency, except that the curves on $Glass015vs2$, $Abalone19$ and $Yeast1289vs7$ decline obviously when λ=0.3. On the whole, the best performance in Fig. 4 is usually achieved when λ is between 0.2 and 0.4. We argue that the optimum performance is achieved by balancing the undersampling and

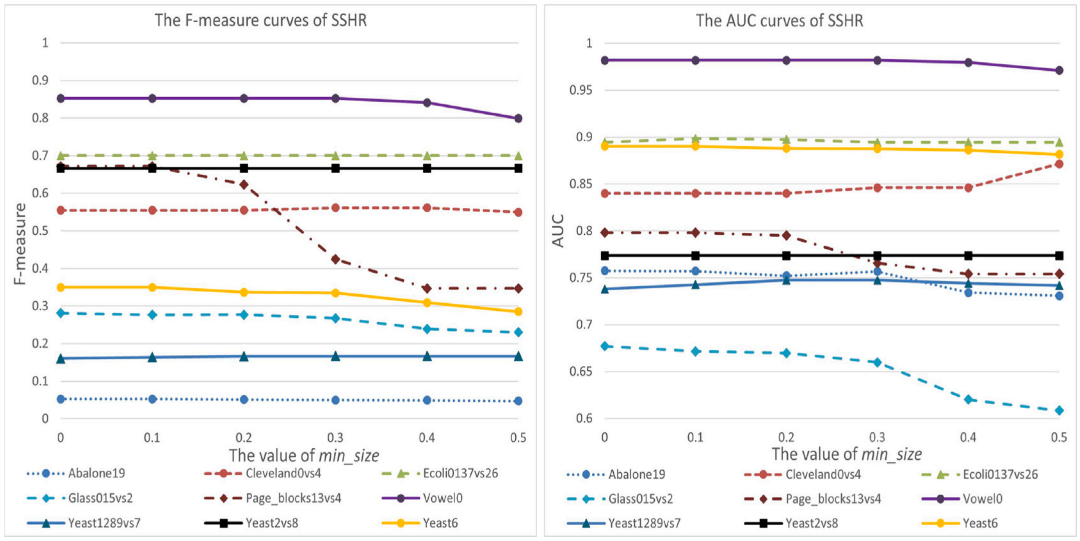


Fig. 3. The impact of the hyperparameter min_size on SSHR.

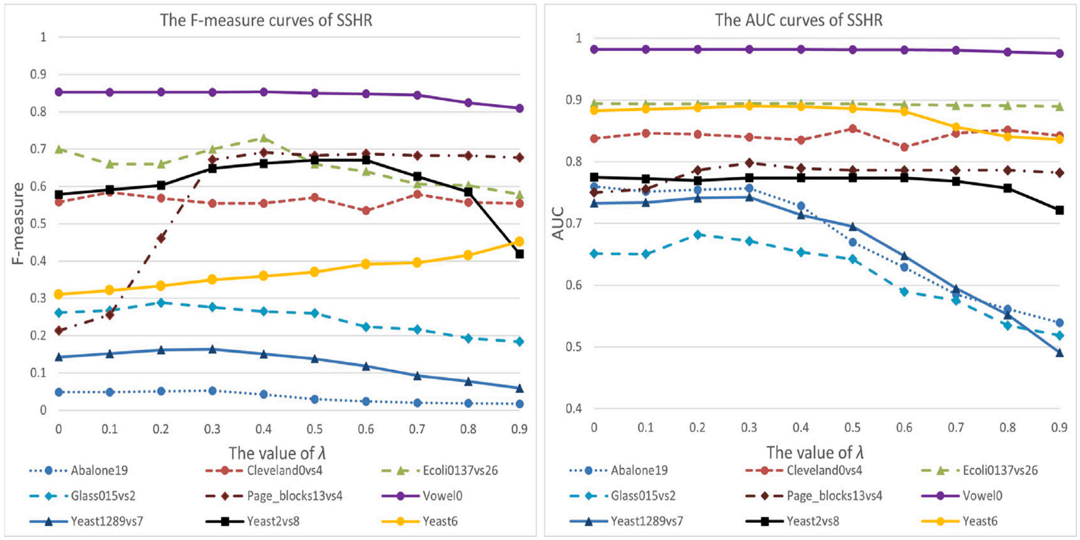


Fig. 4. The impact of the hyperparameter λ on SSHR.

Table 8
The result of ablation study.

Method	F-measure	AUC	F-measure-ranking	AUC-ranking
Original svm	0.251	0.614	4.989	5.284
SSHR-Unlabeled	0.463	0.754	3.000	3.171
SSHR-SSHC	0.367	0.779	4.159	3.364
SSHR-Under	0.592	0.839	2.830	2.886
SSHR-Over	0.322	0.657	4.568	4.932
SSHR	0.644	0.862	1.455	1.364

oversampling ratios. Note that undersampling might delete informative samples while oversampling has the risk of incorporating noise. These phenomena in Fig. 4 illustrate the benefit of leveraging the strengths of both resampling approaches.

4.3.4. *An ablation study*

This paper proposes a semi-supervised hierarchical clustering algorithm, SSHC, to capture the class distribution. Based on the result of SSHC, we design a hybrid oversampling and undersampling technique for CIL. We perform an ablation study to analyze the contribution of each critical technique in SSHR. Table 8 shows the result of the ablation study, where SSHR-Over, SSHR-Under and SSHR-Unlabeled denote the result of removing oversampling, undersampling and the usage of unlabeled data from SSHR, respectively. SSHR-SSHC denotes the result of replacing SSHC with a traditional semi-supervised clustering algorithm, Seeded Kmeans (Basu et al., 2002)

The effect of SSHC

Unlike those unsupervised clustering used in existing resampling methods, SSHC learns from both labeled and unlabeled data to capture the class distribution. To evaluate the benefit of utilizing unlabeled data, we first run SSHC on only labeled data. As shown in Table 8, the performances of SSHR-Unlabeled data decrease by 18.1% and 10.8%, in terms of F-measure and AUC, respectively. Next, we replace

SSHC with Seeded Kmeans to show the superiority of SSHC. Measured by F-measure and AUC, SSHR-SSHC performs worse than SSHR by 27.7% and 8.3%, respectively. It demonstrates the strength of the cluster-splitting and the objective function used in SSHC. Nevertheless, SSHR-SSHC performs much better than SVM, showing the benefit of incorporating semi-supervised clustering into CIL.

The effect of oversampling/undersampling

To evaluate the strength of the proposed undersampling technique, we remove Step 2 from Algorithm 2. It is observed that the performance of SSHR decreases by 5.2% and 2.3%, in terms of F-measure and AUC, respectively. The impact of undersampling is less significant than other techniques. The main reason, we argue, is that SSHR employs a rigorous strategy to control the number of undersampling to avoid information loss.

By contrast, we remove Step 3 from Algorithm 2 to evaluate the effect of the oversampling technique. We observed that the performance of SSHR drops sharply in Table 8. It shows that the proposed oversampling technique has more effect than undersampling in SSHR. This result further demonstrates the benefit of utilizing unlabeled data in CIL.

5. Conclusion

In this paper, we have proposed a hybrid resampling method, called SSHR, which designs a semi-supervised clustering algorithm to assist the undersampling and oversampling processes. Unlike existing clustering-based resampling methods, the proposed method takes advantage of unlabeled data to disclose the data distribution and enlarge the training set of the minority class. In addition to addressing the class imbalance problem, it alleviates the insufficiency of labeled data.

The effectiveness of the proposed method has been validated on 44 benchmark datasets. With the assistance of semi-supervised clustering, the proposed method shows significant superiority over nine state-of-the-art resampling methods in terms of the average values of F-measure and AUC. The Friedman test further demonstrates that SSHR outperforms the other algorithms significantly. These results show that taking advantage of semi-supervised clustering does improve the effectivity of resampling methods on class-imbalanced datasets. Furthermore, the computational complexity analysis shows that semi-supervised clustering can decrease the time-consuming of resampling, although the incorporation of unlabeled data brings additional computational costs.

The undersampling/oversampling ratio plays a vital role in resampling methods. How to find an optimum ratio remains a challenging issue in CIL. It has been proved for classification tasks that a perfectly balanced distribution does not always provide an optimal result. In further work, we will carry out research on utilizing transductive learning, together with some prior knowledge, to find the optimum resampling ratio in class-imbalanced problems.

CRediT authorship contribution statement

Zhen Jiang: Investigation, Conceptualization, Methodology, Writing – original draft. **Lingyun Zhao:** Data processing, Software, Writing – editing. **Yu Lu:** Visualization, Validation. **Yongzhao Zhan:** Supervision, Writing – reviewing. **Qirong Mao:** Writing – reviewing, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The source code and related datasets are available at https://codeocean.com/capsule/5051082/tree/v1.

Acknowledgments

This research is supported by the National Natural Science Foundation of China (NSFC: 62176106) and the Jiangsu key research and development plan (industry foresight and key core technology, BE2020036).

References

Anand, S., Mittal, S., Tuzel, O., & Meer, P. (2013). Semi-supervised kernel mean shift clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *36*, 1201–1215.

Barua, S., Islam, M. M., Yao, X., & Murase, K. (2012). Mwmote–majority weighted minority oversampling technique for imbalanced data set learning. *IEEE Transactions on Knowledge and Data Engineering*, *26*, 405–425.

Bashir, S., Doolan, D., & Petrovski, A. (2015). Clusternn: A hybrid classification approach to mobile activity recognition. In *Proceedings of the 13th international conference on advances in mobile computing and multimedia* (pp. 263–267).

Basu, S., Banerjee, A., & Mooney, R. (2002). Semi-supervised clustering by seeding. In *Proceedings of 19th international conference on machine learning*. Citeseer.

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*, 20–29.

Bej, S., Davtyan, N., Wolfien, M., Nassar, M., & Wolkenhauer, O. (2021). Loras: An oversampling approach for imbalanced datasets. *Machine Learning*, *110*, 279–301.

Bunkhumpornpat, C., Sinapiromsaran, K., & Lursinsap, C. (2009). Safe-level-smote: Safe-level-synthetic minority over-sampling technique for handling the class imbalanced problem. In *Pacific-asia conference on knowledge discovery and data mining* (pp. 475–482). Springer.

Bunkhumpornpat, C., Sinapiromsaran, K., & Lursinsap, C. (2012). Dbsmote: Density-based synthetic minority over-sampling technique. *Applied Intelligence*, *36*, 664–684.

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). Smote: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, *16*, 321–357.

Chen, C., Qian, H., Chen, W., Zheng, Z., & Zhu, H. (2019). Auto-weighted multi-view constrained spectral clustering. *Neurocomputing*, *366*, 1–11.

Douzas, G., & Bacao, F. (2019). Geometric smote a geometrically enhanced drop-in replacement for smote. *Information Sciences*, *501*, 118–135.

Douzas, G., Bacao, F., & Last, F. (2018). Improving imbalanced learning through a heuristic oversampling method based on k-means and smote. *Information Sciences*, *465*, 1–20.

Elkan, C. (2001). The foundations of cost-sensitive learning. In *International joint conference on artificial intelligence* (pp. 973–978). Lawrence Erlbaum Associates Ltd.

Elyan, E., Jamieson, L., & Ali-Gombe, A. (2020). Deep learning for symbols detection and classification in engineering drawings. *Neural Networks*, *129*, 91–102.

Engelmann, J., & Lessmann, S. (2021). Conditional wasserstein gan-based oversampling of tabular data for imbalanced learning. *Expert Systems with Applications*, *174*, Article 114582.

Galar, M., Fernández, E., & Herrera, F. (2013). Eusboost: Enhancing ensembles for highly imbalanced data-sets by evolutionary undersampling. *Pattern Recognition*, *46*, 3460–3471.

García, S., Fernández, A., Luengo, J., & Herrera, F. (2010). Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power. *Information Sciences*, *180*, 2044–2064.

García, S., & Herrera, F. (2009). Evolutionary undersampling for classification with imbalanced datasets: Proposals and taxonomy. *Evolutionary Computation*, *17*, 275–306.

Gertrudes, J. C., Zimek, A., Sander, J., & Campello, R. J. (2018). A unified framework of density-based clustering for semi-supervised classification. In *Proceedings of the 30th international conference on scientific and statistical database management* (pp. 1–12).

Han, H., Wang, W. Y., & Mao, B. H. (2005). Borderline-smote: A new over-sampling method in imbalanced data sets learning. In *International conference on intelligent computing* (pp. 878–887). Springer.

Hao, J., Wang, C., Yang, G., Gao, Z., Zhang, J., & Zhang, H. (2021). Annealing genetic gan for imbalanced web data learning. *IEEE Transactions on Multimedia*, *24*, 1164–1174.

He, H., Bai, Y., Garcia, E., & Li, S. A. (2008). Adaptive synthetic sampling approach for imbalanced learning. In *2008 (IEEE world congress on computational intelligence), Ieee international joint conference on neural networks*.

Hoyos-Osorio, J., Alvarez-Meza, A., Daza-Santacoloma, G., Orozco-Gutierrez, A., & Castellanos-Dominguez, G. (2021). Relevant information undersampling to support imbalanced data classification. *Neurocomputing*, *436*, 136–146.

Iranmehr, A., Masnadi-Shirazi, H., & Vasconcelos, N. (2019). Cost-sensitive support vector machines. *Neurocomputing*, *343*, 50–64.

Jiang, Z., Zhan, Y., Mao, Q., & Du, Y. (2022). Semi-supervised clustering under a compact-cluster assumption. *IEEE Transactions on Knowledge and Data Engineering*.

Jo, T., & Japkowicz, N. (2004). Class imbalances versus small disjuncts. *ACM Sigkdd Explorations Newsletter*, *6*, 40–49.

Kang, Q., Chen, X., Li, S., & Zhou, M. (2016). A noise-filtered under-sampling scheme for imbalanced classification. *IEEE Transactions on Cybernetics*, *47*, 4263–4274.

Kaur, H., Pannu, H. S., & Malhi, A. K. (2019). A systematic review on imbalanced data challenges in machine learning: Applications and solutions. *ACM Computing Surveys*, *52*, 1–36.

Kubat, M., Matwin, S., et al. (1997). Addressing the curse of imbalanced training sets: One-sided selection. In *Icml* (p. 179). Citeseer.

Kumar, S., Biswas, S. K., & Devi, D. (2019). Tlusboost algorithm: A boosting solution for class imbalance problem. *Soft Computing*, *23*, 10755–10767.

Lai, Y., He, S., Lin, Z., Yang, F., Zhou, Q., & Zhou, X. (2019). An adaptive robust semi-supervised clustering framework using weighted consensus of random k k-means ensemble. *IEEE Transactions on Knowledge and Data Engineering*, *33*, 1877–1890.

Lim, P., Goh, C. K., & Tan, K. C. (2016). Evolutionary cluster-based synthetic oversampling ensemble (eco-ensemble) for imbalance learning. *IEEE Transactions on Cybernetics*, *47*, 2850–2861.

Lin, W. C., Tsai, C. F., Hu, Y. H., & Jhang, J. S. (2017). Clustering-based undersampling in class-imbalanced data. *Information Sciences*, *409*, 17–26.

Liu, Z., Cao, W., Gao, Z., Bian, J., Chen, H., Chang, Y., et al. (2020). Self-paced ensemble for highly imbalanced massive data classification. In *2020 IEEE 36th international conference on data engineering* (pp. 841–852). IEEE.

Liu, H., Tao, Z., & Fu, Y. (2017). Partition level constrained clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *40*, 2469–2483.

Lu, H., Chen, C., Wei, H., Ma, Z., Jiang, K., & Wang, Y. (2022). Improved deep convolutional embedded clustering with re-selectable sample training. *Pattern Recognition*, *127*, Article 108611.

Mirzaei, B., Nikpour, B., & Nezamabadi-pour, H. (2021). Cdbh: A clustering and density-based hybrid approach for imbalanced data classification. *Expert Systems with Applications*, *164*, Article 114035.

Mullick, S. S., Datta, S., & Das, S. (2018). Adaptive learning-based k-nearest neighbor classifiers with resilience to class imbalance. *IEEE Transactions on Neural Networks and Learning Systems*, *29*, 5713–5725.

Nekooeimehr, I., & Lai-Yuen, S. K. (2016). Adaptive semi-unsupervised weighted oversampling (a-Suwo) for imbalanced datasets. *Expert Systems with Applications*, *46*, 405–416.

Ofek, N., Rokach, L., Stern, R., & Shabtai, A. (2017). Fast-cbus: A fast clustering-based undersampling method for addressing the class imbalance problem. *Neurocomputing*, *243*, 88–102.

Peng, M., Zhang, Q., Xing, X., Gui, T., Huang, X., Jiang, Y. G., et al. (2019). Trainable undersampling for class-imbalance learning. In *Proceedings of the AAAI conference on artificial intelligence* (pp. 4707–4714).

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*, 245–265.

Ren, Y., Hu, X., Shi, K., Yu, G., Yao, D., & Xu, Z. (2018). Semi-supervised denpeak clustering with pairwise constraints. In *Pacific rim international conference on artificial intelligence* (pp. 837–850). Springer.

Richhariya, B., & Tanveer, M. (2020). A reduced universum twin support vector machine for class imbalance learning. *Pattern Recognition*, *102*, Article 107150.

Seiffert, C., Khoshgoftaar, T. M., Van Hulse, J., & Napolitano, A. (2010). Rusboost: A hybrid approach to alleviating class imbalance. *IEEE Transactions on Systems Man and Cybernetics—Part A: Systems and Humans*, *40*(185).

Sun, J., Lang, J., Fujita, H., & Li, H. (2018). Imbalanced enterprise credit evaluation with dte-sbd: Decision tree ensemble based on smote and bagging with differentiated sampling rates. *Information Sciences*, *425*, 76–91.

Tao, X., Li, Q., Guo, W., Ren, C., He, Q., Liu, R., et al. (2020). Adaptive weighted over-sampling for imbalanced datasets based on density peaks clustering with heuristic filtering. *Information Sciences*, *519*, 43–73.

Tao, D., Tang, X., Li, X., & Wu, X. (2006). Asymmetric bagging and random subspace for support vector machines-based relevance feedback in image retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *28*, 1088–1099.

Tomek, I. (1976). Two modifications of cnn.

Tsai, C. F., Lin, W. C., Hu, Y. H., & Yao, G. T. (2019). Under-sampling class imbalanced datasets by combining clustering analysis and instance selection. *Information Sciences*, *477*, 47–54.

Van Engelen, J. E., & Hoos, H. H. (2020). A survey on semi-supervised learning. *Machine Learning*, *109*, 373–440.

Vu, V. V., & Do, H. Q. (2017). Density-based clustering with side information and active learning. In *2017 9th international conference on knowledge and systems engineering* (pp. 166–171). IEEE.

Vuttipittayamongkol, P., & Elyan, E. (2020). Neighbourhood-based undersampling approach for handling imbalanced and overlapped data. *Information Sciences*, *509*, 47–70.

Wagstaff, K., Cardie, C., Rogers, S., Schrödl, S., et al. (2001). Constrained k-means clustering with background knowledge. In *Icml* (pp. 577–584).

Wang, Z., Cao, C., & Zhu, Y. (2020). Entropy and confidence-based undersampling boosting random forests for imbalanced problems. *IEEE Transactions on Neural Networks and Learning Systems*, *31*, 5178–5191.

Weiss, G. M. (2004). Mining with rarity: A unifying framework. *ACM Sigkdd Explorations Newsletter*, *6*, 7–19.

Wong, G. Y., Leung, F. H., & Ling, S. H. (2018). A hybrid evolutionary preprocessing method for imbalanced datasets. *Information Sciences*, *454*, 161–177.

Xu, Q., Desjardins, M., & Wagstaff, K. (2005). Constrained spectral clustering under a local proximity structure assumption. In *In FLAIRS conference*. Citeseer.

Yen, S. J., & Lee, Y. S. (2009). Cluster-based under-sampling approaches for imbalanced data distributions. *Expert Systems with Applications*, *36*, 5718–5727.

Yoder, J., & Priebe, C. E. (2017). Semi-supervised k-means++. *Journal of Statistical Computation and Simulation*, *87*, 2597–2608.

Zeng, S., Tong, X., Sang, N., & Huang, R. (2013). A study on semi-supervised fcm algorithm. *Knowledge and Information Systems*, *35*, 585–612.

Zhang, H., & Li, M. (2014). Rwo-sampling: A random walk over-sampling approach to imbalanced data classification. *Information Fusion*, *20*, 99–116.

Zhang, W., Ramezani, R., & Naeim, A. (2019). Wotboost: Weighted oversampling technique in boosting for imbalanced learning. In *2019 IEEE international conference on big data (Big Data)* (pp. 2523–2531). IEEE.