TreeSMOTE: Structure-Aware Data Augmentation for Imbalanced Tabular Learning

Anonymous authors
Paper under double-blind review

Abstract

Class imbalance has been a critical bottleneck in classification problems, undermining a classifier's identification of minority instances. Data augmentation provides an effective solution by oversampling the minority. Extant methods often generate samples through duplication, perturbation, or interpolation, largely relying on the assumption of local smoothness of the data space to ensure synthetic data reliability. Alternatively, generative models are leveraged for data learning and synthesis. However, both approaches encounter significant limitations in tabular data, primarily due to data heterogeneity and smaller sizes. Specifically, in tabular data with heterogeneous (e.g., continuous, categorical, and ordinal) features, local smoothness may not hold, and thus spatial proximity may fail to capture the underlying data distribution, resulting in noisy or misleading synthetic instances. Moreover, generative models can overfit (memorize) smaller tabular data. To bridge the gap in tailored augmentation for imbalanced tabular classification, we propose TreeSMOTE, a structure-aware oversampling framework that shifts the focus from the feature space to a tree-induced decision space. We quantify the similarity between samples based on the depth of their lowest common ancestor in decision paths from trees and synthesize label-consistent instances based on structurally similar samples. TreeSMOTE readily combines with and improves classification models by increasing data diversity and reducing label contamination. Extensive experiments on large-scale imbalanced tabular learning benchmarks demonstrate that TreeSMOTE consistently outperforms popular oversampling methods and generative models and further improves state-of-the-art ensemble approaches. Preserving the data manifold, TreeSMOTE offers an effective and efficient solution in imbalanced tabular data environments.

1 Introduction

Tabular data is one of the most prevalent data formats in real-world applications, such as financial transactions for fraud detection (Kennedy et al., 2025), electronic health records for medical diagnosis (Borisov et al., 2021), and industrial logs for anomaly detection (Ye et al., 2025). Class imbalance is a pervasive issue in tabular datasets, where events of interest occur with significantly lower frequency than majority-class instances (Xiao et al., 2021). Class imbalance poses substantial challenges for classification models, leading to predictions biased toward the majority class and poor recognition of the minority that is often of more practical importance.

A common strategy to mitigate class imbalance is data-level rebalancing, including undersampling and oversampling. Undersampling reduces the number of majority-class instances to balance the dataset, but it may discard informative samples and degrade classification performance (Liu et al., 2025). To preserve the information of the majority, oversampling methods that duplicate or generate synthetic samples for the minority class are widely adopted. Modern generative models can be used, like variational autoencoders (VAE), generative adversarial nets, and diffusion models can be adapted to tubular data synthesis (Xu et al., 2019; Adiputra & Wanchai, 2024; Ren et al., 2025). However, the deep generative models may overfit (memorize) smaller-sized tabular data (e.g., van den Burg & Williams, 2021; Baptista et al., 2025), undermining the synthetic data diversity.

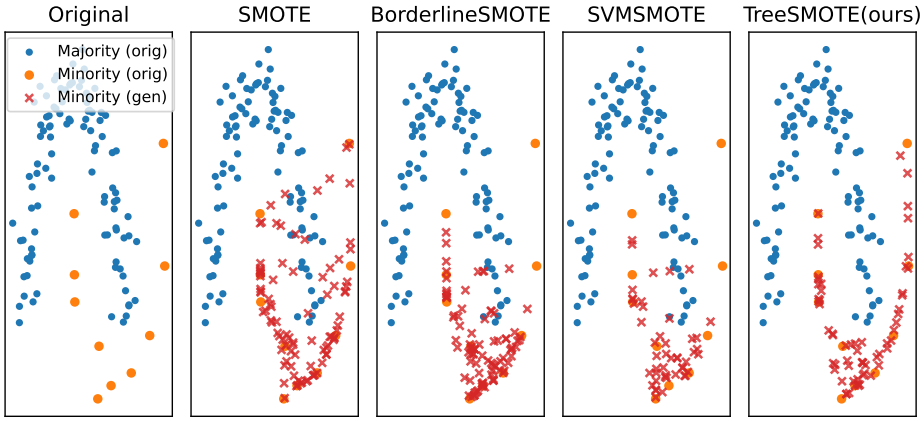


Figure 1: Minority-class augmentation by SMOTE-based methods. The first panel depicts the original data consisting of two crescent moons, with the upper moon (blue dots) being the majority and the lower (orange dots) being the minority. The other panels illustrate synthetic minority instances (red crosses) via SMOTE, BorderlineSMOTE, SVMSMOTE, and TreeSMOTE, respectively. However, TreeSMOTE generates instances for the minority in its region without violating class boundaries.

Without being constrained by the data size, the Synthetic Minority Oversampling Technique (SMOTE) (Chawla et al., 2002) and its variants—such as BorderlineSMOTE (Han et al., 2005) and SVMSMOTE (Nguyen et al., 2011)—are widely adopted for augmenting minority classes. These methods synthesize new instances by interpolating between neighboring samples within the feature space. However, extant SMOTE methods largely relies on local smoothness of the data space between neighboring data points, a premise that is often violated in tabular datasets with heterogeneous (e.g., continuous, categorical, and ordinal) features (Wang et al., 2006). Consequently, the minority data synthesis introduces noisy or misleading instances and degrades classification performance (Fernández et al., 2018), because relying on spatial proximity (e.g., quantified by Euclidean distance) for neighbor selection fails to preserve the underlying data distribution (Feng et al., 2022; Dai et al., 2023).

To resolve this, we shift the perspective from spatial proximity to structural similarity by leveraging tree-induced decisions that partition the feature space into hierarchical, label-homogeneous regions, where local smoothness is better preserved. Samples that share a deeper decision path exhibit a higher structural similarity that transcends their raw feature disparities. We propose *TreeSMOTE*, a structure-aware oversampling framework that constrains data synthesis to decision-consistent regions. To further ensure data quality, a consistency filtering step is introduced to prune unreliable synthetic data with negligible computational overhead. Prioritizing structural similarity over spatial proximity, TreeSMOTE preserves the underlying data distribution and reduces label contamination.

In Figure 1, we illustrate the advantage of TreeSMOTE over SMOTE and its variants, BorderlineSMOTE and SVMSMOTE. These approaches generate minority-class instances violating class regions (i.e., red crosses and blue dots overlap), thereby distorting the original data manifold and undermining downstream classification. In contrast, TreeSMOTE identifies structurally similar neighbors that do not cross class boundaries, ensuring that synthetic samples remain within valid minority regions and preserve the data distribution.

Since tree models are inherently capable of processing nominal features without data transformation, TreeSMOTE is tailored for tabular data, leveraging tree-induced decisions to capture complex feature interdependence and heterogeneous data types—both are critical characteristics of tabular data that may be absent in unstructured data formats. Unlike oversampling that treats features as independent axes in a Euclidean space, TreeSMOTE captures the underlying conditional logic of the data. By measuring how samples traverse hierarchical decision paths, our approach identifies structural similarities that respect class boundaries and feature interactions. We make four primary contributions:

- Simple yet effective oversampling for tabular data. TreeSMOTE is a computationally efficient data augmentation approach that preserves the underlying tabular data distribution.
- Structural similarity via decision paths. We define tree-induced *structural similarity* and *structural distance* based on the depth of the lowest common ancestor in decision paths, offering a distinct and more effective alternative to Euclidean distance for finding proximal neighbors.
- Seamless integration with classification models. TreeSMOTE operates in parallel with imbalanced classification methods and readily integrates with and further improve them.
- Comprehensive empirical validation and consistent improvements. We conduct extensive experiments on large-scale imbalanced tabular benchmarks and demonstrate the consistent improvements of TreeSMOTE, used with a conventional classifier or a tailored model for imbalanced classification.

2 Related Literature

Extensive efforts have been directed toward addressing the challenges of class imbalance. We categorize the extant literature into three primary streams: optimization-based approaches, representation learning, and data-level rebalancing. The proposed TreeSMOTE contributes to data-level rebalancing and can be seamlessly combined with models in the other two streams. We discuss each stream and our contribution to the literature.

2.1 Optimization-Based Approaches

Optimization-based methods do not rely on classifier specification but on training schemes. They address class imbalance that often forces classifiers to favor the majority and overlook the minority (Ye et al., 2024). Rather than optimizing the classifier for an overall performance, these approaches prioritize accuracy across specific groups of interest. A significant body of work adopts distributionally robust optimization (DRO) to improve worst-case performance. For instance, Group-DRO (Sagawa et al., 2019) minimizes the maximum loss across groups rather than the global average. Subsequent research has further refined this by emphasizing the minority during training (Goel et al., 2020; Levy et al., 2020; Sagawa et al., 2020; Zhang et al., 2021; Deng et al., 2023).

Two-stage training has been developed to automatically identify the minority in a data-agnostic manner, minimizing the impacts of group labeling. In the first stage of training, they identify hard-to-classify instances (Nam et al., 2020; Liu et al., 2021; Yenamandra et al., 2023). In the second stage, they focus training on these instances to let the classifier better learn their distributions (Du et al., 2023; Moayeri et al., 2023; Yang et al., 2024). Our proposed TreeSMOTE framework complements existing approaches in this stream through seamlessly integrating with them and providing diversed supplementary samples, thereby facilitating a more robust training process.

2.2 Representation Learning

Representation learning in the presence of class imbalance focuses on capturing invariant features that remain consistently predictive across both the majority and minority (Sun et al., 2021; Veitch et al., 2021; Eastwood et al., 2023) and reducing the influence of spurious correlation between features and labels arising from biased data. Recent studies (Rosenfeld et al., 2022; Zhong et al., 2024) suggest that standard empirical risk minimization learns both spurious and invariant feature representations. This implies that prediction accuracy can be improved by highlighting invariant feature representations and downplaying the spurious.

Building on this, several techniques involve retraining the last layer of a neural network on small, held-out datasets where spurious correlations are broken (Kirichenko et al., 2023; Izmailov et al., 2022). They can be combined with automatic minority identification, such as using disagreements between regular empirical risk minimization and early-stopping (Chen et al., 2023; LaBonte et al., 2023). In addition, regularization has been used to minimize the classifier's learning of spurious correlation. Specifically, using nuclear-norm

penalty (Shi et al., 2024) or both nuclear- and Frobenius-norm penalties (Wen et al., 2025) exhibits similar properties to LASSO or elastic net to emphasize invariant feature representations.

2.3 Data-Level Rebalancing

Data-level interventions balance classes before training a classifier. This stream of work can be categorized into oversampling and undersampling. Oversampling, or data augmentation, seeks to expand the minority class by generating new instances rather than simple duplication. As one of the most popular augmentation schemes, SMOTE (Chawla et al., 2002) generates synthetic samples via interpolation between minority samples. To address potential noise and overlapping class boundaries, variants such as BorderlineSMOTE (Han et al., 2005) and SVMSMOTE (Nguyen et al., 2011) focus on generation in critical decision regions or support vectors, and Batista et al. (2004) propose to remove ambiguous instances using Tomek links or nearest neighbor rules. For tabular data generation, generative models can be leveraged, like variational autoencoders, generative adversarial nets, and diffusion models (Xu et al., 2019; Adiputra & Wanchai, 2024; Ren et al., 2025), but their training necessitate larger datasets and often impose significant computational demands.

Undersampling adjusts class proportions through removing majority instances but risks discarding informative samples. To address this limitation, ensemble-based approaches that combine base learners trained on undersampled, balanced data have become a prominent solution (Zhao et al., 2023). For example, diversity can be enhanced by training multiple classifiers on distinct undersampled majority subsets (Barandela et al., 2003), and performance is further improved by gradually focusing on hard-to-classify instances (Liu et al., 2008). Self-Paced Ensemble (Liu et al., 2020) considers classification difficulty by assigning higher sampling weights to challenging majority-class instances and has achieved state-of-the-art performance. Other representative methods, such as RUSBoost (Seiffert et al., 2009) and Balanced Random Forest (Khoshgoftaar et al., 2007), combine undersampling with boosting or bootstrap aggregation to enhance minority-class recall.

The proposed TreeSMOTE contributes to data-level rebalancing approaches. As a data augmentation tool, it shifts the data synthesis from the feature space to a tree-induced decision space, ensuring that synthetic samples respect the underlying data distribution. It further reduces noisy synthesis through a consistency filtering step without incurring additional training cost. More importantly, TreeSMOTE operates in parallel and readily combines with optimization-based approaches, representation learning, and undersampling ensembles. Providing diverse, structure-aware synthetic samples, it further improves established approaches and allows for more robust learning of imbalanced data.

3 Method

We first provide notations and preliminary knowledge of SMOTE and classification tree. Then, we define tree-induced structural similarity and distance based on the concept of lowest common ancestor and formally introduce the TreeSMOTE algorithm. In addition, since TreeSMOTE conducts data-level rebalancing and operates in parallel with classification approaches, we show as an example how TreeSMOTE combines with undersampling ensembles, pushing the boundaries of their established state-of-the-art performance (Liu et al., 2025).

3.1 Preliminaries

3.1.1 Imbalanced Data Classification

We consider a classification problem on imbalanced tabular data $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$, where the feature $\boldsymbol{x}_i$ can be real and/or nominal valued, and the class label $y_i \in \{1, \ldots, C\}$. The dataset is imbalanced, with the majority and minority classes $\mathcal{D}_{\text{maj}}$ and $\mathcal{D}_{\text{min}}$ such that $\mathcal{D} = \mathcal{D}_{\text{maj}} \cup \mathcal{D}_{\text{min}}$ and $|\mathcal{D}_{\text{maj}}| \gg |\mathcal{D}_{\text{min}}|$, where $|.|$ denotes the cardinality. The class imbalance ratio (IR) is defined as $\text{IR} = |\mathcal{D}_{\text{maj}}|/|\mathcal{D}_{\text{min}}|$. A higher IR indicates a more severe imbalance, which often biases classifiers toward the majority class.

3.1.2 SMOTE Overview

The synthetic minority oversampling technique (SMOTE) is a widely used to generate synthetic samples, often for minority classes, via interpolation between existing instances. Given a sample $\boldsymbol{x}_i$ and one of its spatially nearest neighbors $\boldsymbol{x}_j$ in the same class, a new sample is generated as

$$\boldsymbol{x}_i^{\text{syn}} = \boldsymbol{x}_i + \lambda(\boldsymbol{x}_j - \boldsymbol{x}_i), \quad \lambda \sim \text{Uniform}(0, 1). \tag{1}$$

Extant SMOTE approaches largely encode nominal features into one-hot vectors and apply Equation equation 1 without distinguishing from real-valued features. SMOTE increases the number of minority samples and mitigates class imbalance, but may generate misleading samples, particularly when local smoothness does not hold, and thus interpolations do not align with the underlying data distribution (e.g., Figure 1).

3.1.3 Trees and Hierarchical Structures

A classification and regression tree predicts a target label by learning decision rules inferred from features. The model is structured as a binary tree, where each internal node represents a question about a specific input feature, and each branch represents the outcome of that test. The process begins at the root node, where the algorithm selects the feature and specific split point that most effectively partitions the data into more homogeneously labeled subsets. For regression trees where the label is continuous, the purity of a node is usually based on minimizing the mean squared error. Tree models are more commonly used for classification problems, where the purity can be measured by the Gini index or information gain (Loh, 2011).

The recursive partitioning continues down the tree until a stopping criterion is met, such as a maximum depth or a minimum number of samples, at which point the final leaf nodes provide the predicted continuous or categorical label. Features chosen near the root typically have higher importance for classification, whereas deeper splits often involve less discriminative features. Each sample follows a path from the root to a leaf determined by these hierarchical decisions. For a given sample $\boldsymbol{x}_i$, its decision path is $\pi(\boldsymbol{x}_i) = \big(v_{i,1}, v_{i,2}, \ldots, v_{i,L_i}\big)$, where $v_{i,k}$ denotes the k-th visited node, and L_i is the depth of the leaf reached by $\boldsymbol{x}_i$.

3.2 TreeSMOTE for Structure-Aware Data Augmentation

3.2.1 LCA Depth and Similarity

To quantify structural similarity with respect to feature importance and dependence, we leverage *lowest common ancestor* (LCA), which can be defined in tree models as the deepest ancestor of two nodes (Harel & Tarjan, 1984; Bender & Farach-Colton, 2000). We define the *LCA depth* for two samples $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ along their decision paths in a tree as

$$\text{Depth}_{\text{LCA}}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \max\big\{\ell : v_{i,k} = v_{j,k},\ \forall k = 1, \ldots, \ell\big\}, \tag{2}$$

where $v_{i,k}$ and $v_{j,k}$ denote the k-th nodes visited by $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ along their respective paths $\pi(\boldsymbol{x}_i)$ and $\pi(\boldsymbol{x}_j)$ from the root.

A larger $\text{Depth}_{\text{LCA}}(\boldsymbol{x}_i, \boldsymbol{x}_j)$ indicates that the two samples share a longer decision path along features starting from the root, which are often of higher importance in label prediction than those far from the root, and they remain indistinguishable by earlier splits of the tree. Therefore, $\text{Depth}_{\text{LCA}}$ provides a structure-aware measure of similarity in the partitioned feature space (or equivalently, the decision space) induced by the tree, and a larger value indicates a higher degree of similarity.

While a tree defines a hierarchical partition of the feature space, it is prone to overfit, sensitive to specific features, and of high variance (Loh, 2011). To addess the limitations, we consider multiple trees, $\{Tree^{(t)}\}_{t=1}^{T}$, where each tree is trained on a random subset of features, to facilitate the learning of various hierarchies of the feature space and provide a more robust and comprehensive measure of structural similarity. Note that this is different from random forests as we do not use boostrap samples. Specifically, we define the

tree-induced *structural similarity* between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ across T trees as

$$S(\boldsymbol{x}_i, \boldsymbol{x}_j) = \frac{1}{T} \sum_{t=1}^{T} \mathrm{Depth_{LCA}}^{(t)}(\boldsymbol{x}_i, \boldsymbol{x}_j), \tag{3}$$

where $\mathrm{Depth_{LCA}}^{(t)}(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is calculated by Equation equation 2 based on tree t. Consequently, we define the tree-induced *structural distance* as

$$D(\boldsymbol{x}_i, \boldsymbol{x}_j) = H - S(\boldsymbol{x}_i, \boldsymbol{x}_j), \tag{4}$$

where H is the maximum path length (i.e., the maximum tree depth). A larger structural distance indicates that the two samples' decision paths diverge earlier. The structural distance in Equation equation 4 that integrates information from multiple trees comprehensively captures feature hierarchies and dependence, providing a more reliable measure of structural proximity between samples. The following proposition establishes the ultrametric property of the structural distance, which is absent in spatial proximity metrics like Euclidean distance.

proposition[Ultrametric property of one-tree structural distance] Let $H^{(t)}$ be the depth of $Tree^{(t)}$. The tree-induced structural distance $D^{(t)}(\boldsymbol{x}_i, \boldsymbol{x}_j) = H^{(t)} - \mathrm{Depth_{LCA}}^{(t)}(\boldsymbol{x}_i, \boldsymbol{x}_j)$ satisfies the ultrametric inequality $D^{(t)}(\boldsymbol{x}_i, \boldsymbol{x}_k) \leq \max\left(D^{(t)}(\boldsymbol{x}_i, \boldsymbol{x}_j), D^{(t)}(\boldsymbol{x}_j, \boldsymbol{x}_k)\right)$, for any $\boldsymbol{x}_i$, $\boldsymbol{x}_j$, and $\boldsymbol{x}_k$.

The ultrametric property distinguishes the tree-induced structural distance from spatial proximity. Specifically, instances with a short Euclidean distance are geometrically close but can diverge early along a decision path. In contrast, a smaller structural distance indicates that the two instances share longer decision paths across the T trees. Since a tree splits nodes by reducing node impurity, leading to increasingly homogeneous label distributions in deeper nodes, two instances with a shorter structural distance are within a more refined and label-homogeneous region where local smoothness is better preserved.

3.2.2 Structure-Aware Data Synthesis

We formally introduce synthetic data generation by TreeSMOTE. For each (minority) instance $\boldsymbol{x}_i$, we first select at random one of its k structurally similar samples of the same class, where the distance is measured by $D(\boldsymbol{x}_i, .)$ Then, synthetic samples can be generated by Equation equation 1 using the sample x_i and one of its structurally similar samples x_j. The data generation process is repeated until a desired number of synthetic samples is obtained.

Nearest neighbors quantified by the tree-induced structural distance tend to be in a label-homogeneous region, where local smoothness of data space is better preserved. Restricting the synthesis using structurally similar samples, TreeSMOTE reduces the risk of generating noisy instances and preserves the intrinsic interdependence among features. By generating synthetic samples from structurally nearest neighbors, TreeSMOTE promotes label consistency and reduces label noise. Our empirical evaluation shows that a one-tree configuration of TreeSMOTE (i.e., $T = 1$) can already outperform extant SMOTE variants. Notably, performance gains scale positively with the number of trees until T reaches the range of 50 to 100. The training of trees is computationally efficient and can be further accelerated through parallel computing.

3.2.3 Consistency Filtering

To further improve the reliability of the data augmentation, each synthetic instance $\boldsymbol{x}_i$ is evaluated by $\{Tree^{(t)}\}_{t=1}^{T}$ without re-training:

$$\hat{y}_i^{\mathrm{syn}} = \mathrm{mode}\Big(\{\hat{y}^{(t)}(\boldsymbol{x}_i^{\mathrm{syn}})\}_{t=1}^{T}\Big), \tag{5}$$

where $\hat{y}^{(t)}(\boldsymbol{x}_i^{\mathrm{syn}})$ is the predicted label for $\boldsymbol{x}_i^{\mathrm{syn}}$ by $Tree^{(t)}$, and mode$(\cdot)$ returns the most frequently predicted class among all the T trees (i.e., majority voting). The synthetic instance is retained only if $\hat{y}_i^{\mathrm{syn}} = y_i$, that is, its predicted label matches the intended class. This step eliminates instances that cross the decision boundaries induced by the T trees, further mitigating noisy data generation. Note that SMOTE-TomekLinks

and SMOTE-ENN use Tomek Links or an edited nearest neighbor rule to remove ambiguous synthetic data (Batista et al., 2004) but incur extra computational burden. In contrast, the consistency filtering step of TreeSMOTE utilizes the trained trees and requires negligible testing overhead.

To wrap up, TreeSMOTE consists of a data generation step and a consistency filtering step. The data generation step uses samples with a short structural distance to general structurally proximal synthetic data. The consistency filtering step eliminates misleading instances without introducing extra training costs, further reducing noise in the generated data. The implementation of TreeSMOTE is summarized as Algorithm 1.

Algorithm 1 TreeSMOTE for Minority Augmentation

Input: $\mathcal{D} = \mathcal{D}_{maj} \cup \mathcal{D}_{min}$, k, T, N_{gen}
Output: $\tilde{\mathcal{D}}_{min}$
$\tilde{\mathcal{D}}_{min} \leftarrow$ null set
Train $\{\text{Tree}^{(t)}\}_{t=1}^{T}$ on $\mathcal{D}$
while $|\tilde{\mathcal{D}}_{min}| < N_{gen}$ **do**
  Sample $\boldsymbol{x}_i \in \mathcal{D}_{min}$
  Find $\boldsymbol{x}_j$ as one of k structurally nearest neighbor of the same class, where Eq. equation 4 is used as the distance
  Generate $\boldsymbol{x}_i^{\text{syn}}$ via Eq. equation 1
  Find $\hat{y}_i^{\text{syn}}$ by Eq. equation 5
  if $\hat{y}_i^{\text{syn}} = y_i$ **then**
    Add $\{\boldsymbol{x}_i^{\text{syn}}, y_i\}$ to $\tilde{\mathcal{D}}_{min}$
  end if
end while
return $\tilde{\mathcal{D}}_{min}$

Algorithm 2 TreeSMOTE-Augmented Ensemble

Input: $\mathcal{D} = \mathcal{D}_{maj} \cup \mathcal{D}_{min}$, M, untrained base learners $\{f^{(m)}\}_{m=1}^{M}$
Output: Ensemble classifier $f(\cdot)$
for $m = 1$ to M **do**
  Sample a subset $\mathcal{D}_{maj}^{(m)} \subset \mathcal{D}_{maj}$
  Construct a balanced set:

$$\mathcal{D}^{(m)} \leftarrow \mathcal{D}_{maj}^{(m)} \cup \mathcal{D}_{min}$$

  Use TreeSMOTE in Algorithm 1 to augment $\mathcal{D}^{(m)}$ for each class; the augmented data is denoted by $\tilde{D}^{(m)}$
  Train base learner $f^{(m)}$ on $\tilde{D}^{(m)}$
end for
Find the ensemble $f(\cdot)$ by Eq. equation 6
return $f(\cdot)$

3.3 Integrating TreeSMOTE with Undersampling Ensemble

TreeSMOTE as a data-level rebalancing framework is readily integrated with classificaiton approaches. We show as an example how TreeSMOTE combines with ensemble methods.

3.3.1 Undersampling Ensembles

Undersampling ensemble methods are widely used for imbalanced classification. Instead of training a single classifier on the full imbalanced dataset, ensembles construct multiple balanced training subsets by iteratively and randomly reducing samples from the majority class. Specifically, in each iteration $m = 1, \ldots, M$, a subset of majority samples $\mathcal{D}_{\text{maj}}^{(m)} \subset \mathcal{D}_{\text{maj}}$ is randomly selected and combined with the minority samples to form a training subset $\mathcal{D}^{(m)} = \mathcal{D}_{\text{maj}}^{(m)} \cup \mathcal{D}_{\text{min}}$. To ensure balanced labels, the subset is constructed such that $|\mathcal{D}_{\text{maj}}^{(m)}| = |\mathcal{D}_{\text{min}}|$.

A base classifier $f^{(m)}$ is trained on each balanced subset $\mathcal{D}^{(m)}$, and the final prediction is obtained by aggregating the outputs of all the base classifiers:

$$f(x) = \text{Agg}\big(f^{(1)}(x), f^{(2)}(x), \ldots, f^{(M)}(x)\big), \tag{6}$$

where $\text{Agg}(\cdot)$ denotes an ensemble aggregation function such as majority voting or weighted averaging. Utilizing different majority subsets, the undersampling ensembles removes the dominance of the majority class and achieve diversity among base learners.

3.3.2 TreeSMOTE-Augmented Ensemble

Undersampling ensembles rebalance data by using subsets of the majority and suffer from the limitations of data size and diversity. TreeSMOTE can be used in this context to address the limitations. For each

undersampled training subset $\mathcal{D}^{(m)}$, synthetic data are generated for each class (majority and minority), and the TreeSMOTE-augmented subset of $\mathcal{D}^{(m)}$, denoted by $\tilde{\mathcal{D}}^{(m)}$, is used to train a base learner $f^{(m)}$. The final ensemble prediction is obtained by Equation equation 6, i.e., aggregation via majority voting or weighted averaging. We summarize this TreeSMOTE-augmented ensemble as Algorithm 2.

4 Experiments and Results

We show the consistent improvements of TreeSMOTE by comparing it against SMOTE-based and generative-model-based augmentation techniques. Furthermore, we demonstrate TreeSMOTE's ability to combine with undersampling ensembles, extending the capabilities of current state-of-the-art approaches for imbalanced tabular learning. Our evaluation spans 83 datasets with diverse imbalance levels, ranging from nearly balanced to extremely skewed with imbalance ratios over 100. In addition, we evaluate TreeSMOTE by using it with various classification models. The outstanding performance demonstrates TreeSMOTE as a robust, universally applicable solution for imbalanced tabular classification.

4.1 Experiment Setup

We evaluate the proposed method on the CLIMB benchmark (Liu et al., 2025), a large-scale collection of 73 datasets with varying levels of class imbalance. The imbalance ratios (IRs) are categorized as low (IR < 5), medium ($5 \leq$ IR < 10), high ($10 \leq$ IR < 50), and extreme (IR ≥ 50). The data we use have sizes varying from a few hundred to several hundred thousand and include 60 binary-class datasets and 13 multi-class datasets. Since the CLIMB benchmark is primarily evaluated using classical imbalance-learning methods and its large scale poses computational challenges for neural network training, we additionally include 10 datasets to facilitate comparisons with generative-model-based approaches. These datasets are adopted from generative-model-based imbalanced learning literature (Ren et al., 2025) and are available in the Python package imbalanced-learn (LemaÃŽtre et al., 2017). We provide detailed information about datasets in Table 4 and Table 5 in Appendix B.1.

To thoroughly evaluate performance, especially on minority classes, we follow Liu et al. (2025; 2020) to assess the methods using multiple metrics that capture different aspects of performance, including the Area Under the Precision–Recall Curve (AUPRC) (Davis & Goadrich, 2006), F1-score (Sokolova et al., 2006), Matthews Correlation Coefficient (MCC) (Boughorbel et al., 2017), and the geometric mean of precision and recall(G-mean) (Powers, 2020). The detailed definitions of these metrics are provided in Appendix B.2.

We compare the proposed TreeSMOTE with a diverse set of baselines. For classical oversampling methods, we compare with widely used SMOTE-based approaches, including SMOTE (Chawla et al., 2002), BorderlineSMOTE (Han et al., 2005), SVMSMOTE (Nguyen et al., 2011), and SMOTE-TomekLinks (Batista et al., 2004). For generative-model-based methods, we consider TVAE, CTGAN (Xu et al., 2019), CTGAN-ENN (Adiputra & Wanchai, 2024), and DGOT (a diffusion model, Ren et al., 2025), which are modern generative models specifically designed for imbalanced tabular learning. For each dataset and method, we use four classifiers: Classification Tree, LightGBM(GBDT) (Ke et al., 2017), multi-layer perceptron (MLP), and LinearSVC. In addition, when integrating our method with ensemble-based approaches, we evaluate its performance with several ensemble methods that have reported state-of-the-art performance for imbalanced tabular learning (Liu et al., 2025), including SelfPacedEnsemble (Liu et al., 2020), BalancedRandomForest (Khoshgoftaar et al., 2007), RUSBoost (Seiffert et al., 2009), UnderBagging (Barandela et al., 2003), and BalanceCascade (Liu et al., 2008). We defer detailed model implementations to Appendix B.

4.2 Results

In the following tables, the best and second-best results are indicated by bold and underlined numbers, respectively. Performance metrics for the proposed TreeSMOTE are highlighted with gray shading. Results represent the mean performance $\pm$ standard deviation across five training-testing partitions. We defer more results to the Appendix.

4.2.1 SMOTE-Based Methods

We first compare our method with SMOTE-based approaches on the CLIMB benchmark, which contains 73 datasets grouped into four imbalance levels: low (28), medium (24), high (15), and extreme (6). Table 1 reports the average performance using AUPRC, BAC, F1-Score, G-mean, and MCC.

The results show that TreeSMOTE achieves consistent improvements across all imbalance levels and all four downstream classifiers. It delivers the best results in most settings. In particular, under highly imbalanced settings, the performance of most extant SMOTE-based methods tends to degrade significantly. This is especially prominent when using simpler downstream models such as Classification Tree and LinearSVC, where synthetic samples generated by conventional SMOTE-based methods may introduce noisy or ambiguous instances that distort the decision boundary. In contrast, TreeSMOTE is able to preserve and even improve classification performance under these challenging scenarios.

Table 1: Performance comparison (mean±sd) among SMOTE-based minority data augmentation. Reported values are averaged over AUPRC, F1-Score, Balanced Accuracy, MCC, and G-mean. The best and second-best results are highlighted in bold and underlined, respectively.

Classifier	**Oversampling Method**	**Imbalance Level**				
		Low	**Medium**	**High**	**Extreme**	**Average**
Classification Tree	None	61.82±2.41	65.99±2.32	51.67±2.20	65.02±3.82	61.37±2.45
	SMOTE	62.54±2.51	65.90±1.98	51.88±2.19	60.07±3.89	61.25±2.38
	BorderlineSMOTE	61.82±2.62	65.97±2.30	51.59±1.73	61.31±3.39	61.04±2.40
	SVMSMOTE	62.13±2.67	66.36±2.49	52.23±1.78	62.30±2.63	61.50±2.42
	SMOTE-TomekLinks	**62.80±2.62**	66.37±2.25	51.42±2.16	60.74±2.40	61.46±2.39
	TreeSMOTE	62.73±2.52	**66.83±2.17**	**52.35±1.59**	**65.17±3.98**	**62.15±2.33**
LinearSVC	None	66.31±2.25	67.07±1.63	46.61±1.55	59.25±1.64	61.93±1.85
	SMOTE	68.46±2.64	69.72±1.77	50.15±1.59	57.59±1.83	64.22±2.07
	BorderlineSMOTE	67.60±2.58	68.40±1.80	49.10±1.60	58.84±2.22	63.34±2.09
	SVMSMOTE	67.98±2.65	69.38±1.81	49.83±1.91	60.63±2.37	64.11±2.20
	SMOTE-TomekLinks	68.57±2.76	68.89±1.71	48.79±1.43	56.13±2.29	63.59±2.10
	TreeSMOTE	**69.10±2.55**	**70.46±1.70**	**50.76±1.71**	**62.82±2.35**	**65.26±2.08**
MLP	None	69.96±2.15	72.41±1.69	57.69±2.17	66.49±3.14	67.96±2.08
	SMOTE	70.97±2.51	72.91±1.72	58.47±2.01	68.68±2.49	68.85±2.14
	BorderlineSMOTE	70.18±2.18	72.81±1.79	58.75±1.86	69.21±2.99	68.62±2.05
	SVMSMOTE	70.64±2.34	73.32±1.80	59.41±2.02	69.09±3.21	69.08±2.17
	SMOTE-TomekLinks	70.99±2.53	72.50±1.70	57.78±2.00	67.95±2.59	68.52±2.15
	TreeSMOTE	**71.37±2.13**	**73.42±1.65**	**59.42±2.05**	**70.28±3.01**	**69.50±2.03**
LightGBM	None	70.63±2.09	74.03±1.87	61.84±2.79	64.30±5.07	69.53±2.40
	SMOTE	71.46±2.39	74.86±1.59	63.40±2.10	71.64±2.85	71.04±2.11
	BorderlineSMOTE	71.44±2.42	74.64±1.52	63.65±1.91	70.97±4.36	70.95±2.18
	SVMSMOTE	71.43±2.43	75.13±1.59	63.70±2.09	71.53±3.37	71.17±2.16
	SMOTE-TomekLinks	**71.62±2.34**	74.97±2.05	62.88±1.98	68.36±3.57	70.77±2.28
	TreeSMOTE	71.26±2.42	**75.02±1.77**	**64.01±2.03**	**73.85±3.68**	**71.32±2.23**

When using a downstream classifier with higher capacity, such as MLP and LightGBM, the baseline performance is already considerably stronger than that of Classification Tree and LinearSVC. While most SMOTE-based methods can provide some performance improvement in this scenario, TreeSMOTE consistently improves the most and achieves the best results, indicating that the samples generated by our method better capture the underlying data structure and benefit downstream learning. Detailed results for other evaluation metrics, including AUPRC, BAC, F1-Score, G-mean, and MCC, are provided in Appendix C.1.

4.2.2 Generative-Model-Based Methods

For a more comprehensive comparison, we evaluate TreeSMOTE against four popular generative models for tabular data generation: TVAE, CTGAN, CTGAN-ENN, and DGOT. We report the average performance across the five evaluation metrics using four downstream classifiers (Classification Tree, LinearSVC, MLP, and LightGBM) on 10 datasets from the Python package *imbalanced-learn* (LemaÃŽtre et al., 2017). The detailed results are reported in Tables 2.

Table 2: Performance comparison (mean±sd) among minority data augmentation by a generative model or TreeSMOTE. Reported values are averaged over AUPRC, F1-Score, Balanced Accuracy, MCC, and G-mean. The best results are highlighted in bold, and the second-best results are underlined.

Dataset	TVAE	CTGAN	CTGAN-ENN	DGOT	TreeSMOTE
optical_digits	89.98±0.53	86.24±0.79	92.63±0.89	<u>93.52±0.81</u>	**93.55±0.85**
satimage	<u>65.77±0.97</u>	64.01±0.56	61.97±1.66	62.87±1.55	**67.42±1.79**
pen_digits	91.32±0.49	91.64±0.29	91.72±0.68	**95.40±0.77**	<u>95.13±0.34</u>
sick_euthyroid	80.35±3.40	73.36±2.54	73.20±5.65	<u>86.46±1.94</u>	**87.17±1.67**
thyroid_sick	78.64±1.80	71.57±2.78	71.97±4.87	<u>81.32±3.38</u>	**86.20±0.64**
coil_2000	<u>53.94±0.82</u>	51.94±1.36	50.74±0.53	51.87±0.39	**54.99±0.45**
wine_quality	**65.48±1.96**	57.70±2.07	60.01±1.73	58.23±1.63	<u>65.35±2.59</u>
letter_img	86.25±1.73	86.49±1.76	88.42±0.70	**94.33±0.43**	<u>92.34±0.40</u>
abalone_19	47.91±0.49	49.96±1.28	<u>50.13±0.45</u>	49.97±0.45	**51.30±2.61**
isolet	87.97±0.69	<u>89.03±0.44</u>	86.65±0.52	88.15±0.57	**90.06±0.64**

The results show that TreeSMOTE achieves at least the second-best performance in all the settings and the best performance in most of the settings, demonstrating its excellent and consistent performance across datasets and downstream classifiers. Among the baseline methods, DGOT performs better than other generative approaches but underperforms TreeSMOTE, though it is specifically designed for oversampling and imbalanced tabular data learning. TVAE and CTGAN achieve moderate performance overall, possibly because they are designed as general-purpose tabular data generators. Furthermore, CTGAN-ENN performs poorly in most settings. One possible reason is that this method is originally designed for applications that may differ from the benchmark data in this paper, limiting its generalizability when applied to diverse imbalanced classification datasets. Detailed results for other metrics are provided in Appendix C.2.

4.2.3 Ensemble-Based Methods

We further combine TreeSMOTE with several undersampling ensemble approaches: SelfPacedEnsemble, BalancedRandomForest, RUSBoost, UnderBagging, and BalanceCascade, which are among the state-of-the-art methods in the imbalanced data learning literature. Experiments are conducted on the CLIMB benchmark. Tables 3 report the results in terms of the average over the five evaluation metrics.

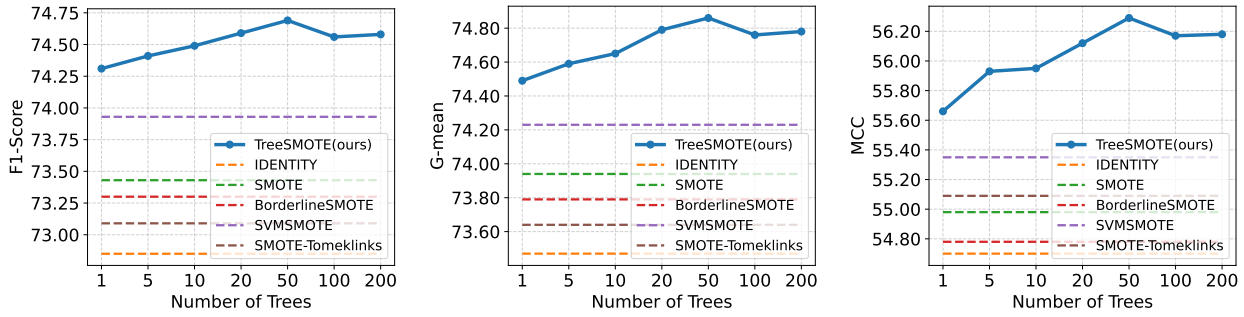
As a result, our method consistently improves the performance of these competitive ensemble approaches in most settings. In addition, the improvement becomes more pronounced as the imbalance ratio increases. This can be attributed to the fact that TreeSMOTE has diversified the sample distribution of each subset constructed by the undersampling ensemble and enhanced the representation of both minority and majority classes. By generating additional samples guided by tree-based partitions, TreeSMOTE helps alleviate the information loss introduced by undersampling while preserving the local decision structure, leading to improved classification performance. Other detailed results are provided in Appendix C.3.

Table 3: Performance comparison averaging over AUPRC, BAC, F1-Score, G-mean, and MCC between ensemble methods and their combinations with TreeSMOTE. Reported values are averaged over AUPRC, F1-Score, Balanced Accuracy, MCC, and G-mean. For each pair, the better result is highlighted in bold.

Method	Low IRs	Medium IRs	High IRs	Extreme IRs	Average
SelfPacedEnsemble	**71.38±2.44**	75.08±1.61	60.98±1.48	71.73±2.44	70.45±1.95
SelfPacedEnsemble w/ TreeSmote	71.28±2.59	**75.32±1.48**	**61.21±1.41**	**71.97±2.67**	**70.57±1.96**
BalancedRandomForest	**71.15±2.57**	73.48±1.44	57.45±1.43	59.01±1.95	67.97±1.89
BalancedRandomForest w/ TreeSmote	70.94±2.50	**73.65±1.62**	**57.88±1.30**	**59.23±1.98**	**68.07±1.90**
RUSBoost	70.46±2.82	72.84±1.58	57.02±1.62	**61.68±2.72**	67.64±2.13
RUSBoost w/ TreeSmote	**70.72±2.66**	**73.45±1.77**	**57.66±1.81**	61.56±2.61	**68.08±2.17**
UnderBagging	**71.09±2.59**	74.16±1.71	58.98±1.18	60.75±2.26	68.66±1.96
UnderBagging w/ TreeSmote	71.05±2.61	**74.17±1.60**	**59.24±1.60**	**61.01±1.83**	**68.73±1.98**
BalanceCascade	70.57±2.45	**74.29±1.58**	56.30±1.76	67.57±2.97	68.53±2.05
BalanceCascade w/ TreeSmote	**71.02±2.61**	74.02±1.72	**56.30±1.61**	**68.02±2.46**	**68.64±2.08**

5 Sensitivity

Impact of the number of trees. Figure 2 presents the average F1-score, G-mean, and MCC of TreeSMOTE combined with an MLP classifier across all datasets in the CLIMB benchmark, as a function of the number of trees.



(a) F1-Score (b) G-mean (c) MCC

Figure 2: Average F1-score, G-mean, and MCC across all datasets with respect to the number of trees, using MLP as the downstream classifier.

Even with a single tree, TreeSMOTE outperforms other SMOTE variants across all three metrics. Performance improves consistently as the number of trees increases, with gains diminishing after approximately 50 trees. These results indicate that while multiple trees are beneficial, a small to moderate ensemble suffices for strong performance. Moreover, tree training is computationally efficient, especially with parallelization.

5.1 Additional Results

We provide additional experimental results in the appendix.

Appendix C.4 presents t-SNE visualizations of the data distributions before and after oversampling on two real-world datasets (*mfeat-zernike* and *Pulsar-Dataset-HTRU2*). Compared with SMOTE, TreeSMOTE

generates synthetic samples that better preserve class boundaries and introduce less noise (see Fig. 3 in the appendix).

Appendix C.5 reports an ablation study evaluating two key components of TreeSMOTE: the tree-induced structural distance versus Euclidean distance, and the consistency filtering step. The results in Table 21 (appendix) demonstrate that both the structural distance and consistency filtering contribute positively to overall performance, with the full configuration achieving the best results across most metrics.

6 Conclusion

We introduce TreeSMOTE, a novel structure-aware oversampling framework designed to address the persistent challenges of class imbalance, particularly within tabular data. While extant SMOTE-based approaches rely on spatial proximity—often failing to capture complex data distributions or the dependencies of heterogeneous features—TreeSMOTE performs data augmentation within a tree-induced decision space. By utilizing the lowest common ancestor depth along decision paths to quantify structural similarity, our method ensures that synthetic samples are generated between structurally compatible instances, better preserving local smoothness and the intrinsic data manifold. Our t-SNE visualizations confirm that shifting to a hierarchical decision space produces high-quality instances that respect class boundaries, resulting in larger-margin class separation and a significant reduction in noisy artifacts.

Extensive empirical evaluations demonstrate that TreeSMOTE consistently delivers superior performance across a wide range of imbalance levels, from nearly balanced to extremely skewed. We have shown that TreeSMOTE is a versatile and computationally efficient framework. It not only functions as an effective standalone data augmentation method but also seamlessly integrates with and improves classification models. In particular, we combine TreeSMOTE with undersampling ensembles and further improve their state-of-the-art performance. Overall, TreeSMOTE provides a fast and effective solution for imbalanced tabular data classification.

References

I Nyoman Mahayasa Adiputra and Paweena Wanchai. Ctgan-enn: a tabular gan-based hybrid sampling method for imbalanced and overlapped data in customer churn prediction. *Journal of Big Data*, 11(1): 121, 2024.

Ricardo Baptista, Agnimitra Dasgupta, Nikola B Kovachki, Assad Oberai, and Andrew M Stuart. Memorization and regularization in generative diffusion models. *arXiv preprint arXiv:2501.15785*, 2025.

Ricardo Barandela, Rosa Maria Valdovinos, and José Salvador Sánchez. New applications of ensembles of classifiers. *Pattern Analysis & Applications*, 6(3):245–256, 2003.

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

Michael A Bender and Martin Farach-Colton. The lca problem revisited. In *Latin American Symposium on Theoretical Informatics*, pp. 88–94. Springer, 2000.

Vadim Borisov, Enkelejda Kasneci, and Gjergji Kasneci. Robust cognitive load detection from wrist-band sensors. *Computers in Human Behavior Reports*, 4:100116, 2021.

Sabri Boughorbel, Fethi Jarray, and Mohammed El-Anbari. Optimal classifier for imbalanced data using matthews correlation coefficient metric. *PloS one*, 12(6):e0177678, 2017.

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

Yongqiang Chen, Wei Huang, Kaiwen Zhou, Yatao Bian, Bo Han, and James Cheng. Understanding and improving feature learning for out-of-distribution generalization. *Advances in Neural Information Processing Systems*, 36, 2023.

Qi Dai, Jian-wei Liu, and Jia-Liang Zhao. Distance-based arranging oversampling technique for imbalanced data. *Neural Computing and Applications*, 35(2):1323–1342, 2023.

Jesse Davis and Mark Goadrich. The relationship between precision-recall and roc curves. In *Proceedings of the 23rd international conference on Machine learning*, pp. 233–240, 2006.

Yihe Deng, Yu Yang, Baharan Mirzasoleiman, and Quanquan Gu. Robust learning with progressive data expansion against spurious correlation. *Advances in neural information processing systems*, 36, 2023.

Yanrui Du, Jing Yan, Yan Chen, Jing Liu, Sendong Zhao, Qiaoqiao She, Hua Wu, Haifeng Wang, and Bing Qin. Less learn shortcut: Analyzing and mitigating learning of spurious feature-label correlation. *IJCAI*, 2023.

Cian Eastwood, Shashank Singh, Andrei L Nicolicioiu, Marin Vlastelica Pogančić, Julius von Kügelgen, and Bernhard Schölkopf. Spuriosity didn't kill the classifier: Using invariant predictions to harness spurious features. *Advances in Neural Information Processing Systems*, 36, 2023.

Shuo Feng, Jacky Keung, Peichang Zhang, Yan Xiao, and Miao Zhang. The impact of the distance metric and measure on smote-based techniques in software defect prediction. *Information and Software Technology*, 142:106742, 2022.

Alberto Fernández, Salvador Garcia, Francisco Herrera, and Nitesh V Chawla. Smote for learning from imbalanced data: progress and challenges, marking the 15-year anniversary. *Journal of artificial intelligence research*, 61:863–905, 2018.

Karan Goel, Albert Gu, Yixuan Li, and Christopher Ré. Model patching: Closing the subgroup performance gap with data augmentation. *arXiv preprint arXiv:2008.06775*, 2020.

Hui Han, Wen-Yuan Wang, and Bing-Huan Mao. Borderline-smote: a new over-sampling method in imbalanced data sets learning. In *International conference on intelligent computing*, pp. 878–887. Springer, 2005.

Dov Harel and Robert Endre Tarjan. Fast algorithms for finding nearest common ancestors. *siam Journal on Computing*, 13(2):338–355, 1984.

Pavel Izmailov, Polina Kirichenko, Nate Gruver, and Andrew G Wilson. On feature learning in the presence of spurious correlations. *Advances in Neural Information Processing Systems*, 35:38516–38532, 2022.

Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, 30, 2017.

Robert KL Kennedy, Flavio Villanustre, and Taghi M Khoshgoftaar. Unsupervised feature selection and class labeling for credit card fraud. *Journal of Big Data*, 12(1):111, 2025.

Taghi M Khoshgoftaar, Moiz Golawala, and Jason Van Hulse. An empirical study of learning from imbalanced data using random forest. In *19th IEEE international conference on tools with artificial intelligence (ICTAI 2007)*, volume 2, pp. 310–317. IEEE, 2007.

Polina Kirichenko, Pavel Izmailov, and Andrew Gordon Wilson. Last layer re-training is sufficient for robustness to spurious correlations. *International Conference on Learning Representations*, 2023.

Tyler LaBonte, Vidya Muthukumar, and Abhishek Kumar. Towards last-layer retraining for group robustness with fewer annotations. *Advances in Neural Information Processing Systems*, 36, 2023.

Guillaume LemaÃŽtre, Fernando Nogueira, and Christos K Aridas. Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning. *Journal of machine learning research*, 18(17): 1–5, 2017.

Daniel Levy, Yair Carmon, John C Duchi, and Aaron Sidford. Large-scale methods for distributionally robust optimization. *Advances in Neural Information Processing Systems*, 33:8847–8860, 2020.

Evan Z Liu, Behzad Haghgoo, Annie S Chen, Aditi Raghunathan, Pang Wei Koh, Shiori Sagawa, Percy Liang, and Chelsea Finn. Just train twice: Improving group robustness without training group information. In *International Conference on Machine Learning*, pp. 6781–6792. PMLR, 2021.

Xu-Ying Liu, Jianxin Wu, and Zhi-Hua Zhou. Exploratory undersampling for class-imbalance learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 39(2):539–550, 2008.

Zhining Liu. Imbens: Python toolbox for class-imbalanced ensemble learning. `https://github.com/ZhiningLiu1998/imbalanced-ensemble`, 2025.

Zhining Liu, Wei Cao, Zhifeng Gao, Jiang Bian, Hechang Chen, Yi Chang, and Tie-Yan Liu. Self-paced ensemble for highly imbalanced massive data classification. In *2020 IEEE 36th international conference on data engineering (ICDE)*, pp. 841–852. IEEE, 2020.

Zhining Liu, Zihao Li, Ze Yang, Tianxin Wei, Jian Kang, Yada Zhu, Hendrik Hamann, Jingrui He, and Hanghang Tong. Climb: Class-imbalanced learning benchmark on tabular data. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2025.

Wei-Yin Loh. Classification and regression trees. *Wiley interdisciplinary reviews: data mining and knowledge discovery*, 1(1):14–23, 2011.

Mazda Moayeri, Wenxiao Wang, Sahil Singla, and Soheil Feizi. Spuriosity rankings: sorting data to measure and mitigate biases. *Advances in Neural Information Processing Systems*, 36:41572–41600, 2023.

Junhyun Nam, Hyuntak Cha, Sungsoo Ahn, Jaeho Lee, and Jinwoo Shin. Learning from failure: De-biasing classifier from biased classifier. *Advances in Neural Information Processing Systems*, 33:20673–20684, 2020.

Hien M Nguyen, Eric W Cooper, and Katsuari Kamei. Borderline over-sampling for imbalanced data classification. *International Journal of Knowledge Engineering and Soft Data Paradigms*, 3(1):4–21, 2011.

David MW Powers. Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation. *arXiv preprint arXiv:2010.16061*, 2020.

Shiqi Ren, Jinliang Ding, and Yiu-ming Cheung. Diffusion gan-based oversampling for imbalanced tabular data. *IEEE Transactions on Knowledge and Data Engineering*, 38(2):983–996, 2025.

Elan Rosenfeld, Pradeep Ravikumar, and Andrej Risteski. Domain-adjusted regression or: Erm may already learn features sufficient for out-of-distribution generalization. *arXiv preprint arXiv:2202.06856*, 2022.

Shiori Sagawa, Pang Wei Koh, Tatsunori B Hashimoto, and Percy Liang. Distributionally robust neural networks for group shifts: On the importance of regularization for worst-case generalization. *International Conference on Learning Representations*, 2019.

Shiori Sagawa, Aditi Raghunathan, Pang Wei Koh, and Percy Liang. An investigation of why overparameterization exacerbates spurious correlations. In *International Conference on Machine Learning*, pp. 8346–8356. PMLR, 2020.

Chris Seiffert, Taghi M Khoshgoftaar, Jason Van Hulse, and Amri Napolitano. Rusboost: A hybrid approach to alleviating class imbalance. *IEEE transactions on systems, man, and cybernetics-part A: systems and humans*, 40(1):185–197, 2009.

Zhenmei Shi, Yifei Ming, Ying Fan, Frederic Sala, and Yingyu Liang. Domain generalization via nuclear norm regularization. In *Conference on Parsimony and Learning*, pp. 179–201. PMLR, 2024.

Marina Sokolova, Nathalie Japkowicz, and Stan Szpakowicz. Beyond accuracy, f-score and roc: a family of discriminant measures for performance evaluation. In *Australasian joint conference on artificial intelligence*, pp. 1015–1021. Springer, 2006.

Xinwei Sun, Botong Wu, Xiangyu Zheng, Chang Liu, Wei Chen, Tao Qin, and Tie-Yan Liu. Recovering latent causal factor for generalization to distributional shifts. *Advances in Neural Information Processing Systems*, 34:16846–16859, 2021.

Gerrit van den Burg and Chris Williams. On memorization in probabilistic deep generative models. *Advances in neural information processing systems*, 34:27916–27928, 2021.

Victor Veitch, Alexander D'Amour, Steve Yadlowsky, and Jacob Eisenstein. Counterfactual invariance to spurious correlations in text classification. *Advances in neural information processing systems*, 34:16196–16208, 2021.

Juanjuan Wang, Mantao Xu, Hui Wang, and Jiwu Zhang. Classification of imbalanced data by using the smote algorithm and locally linear embedding. In *2006 8th international Conference on Signal Processing*, volume 3. IEEE, 2006.

Tao Wen, Zihan Wang, Quan Zhang, and Qi Lei. Elastic representation: Mitigating spurious correlations for group robustness. *Proceedings of The 28th International Conference on Artificial Intelligence and Statistics*, 258:541–549, 2025.

Jin Xiao, Yadong Wang, Jing Chen, Ling Xie, and Jing Huang. Impact of resampling methods and classification models on the imbalanced credit scoring problems. *Information Sciences*, 569:508–526, 2021.

Lei Xu, Maria Skoularidou, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. Modeling tabular data using conditional gan. *Advances in neural information processing systems*, 32, 2019.

Yu Yang, Eric Gan, Gintare Karolina Dziugaite, and Baharan Mirzasoleiman. Identifying spurious biases early in training through the lens of simplicity bias. In *International Conference on Artificial Intelligence and Statistics*, pp. 2953–2961. PMLR, 2024.

Hangting Ye, He Zhao, Wei Fan, Mingyuan Zhou, Dan dan Guo, and Yi Chang. Drl: Decomposed representation learning for tabular anomaly detection. In *The Thirteenth International Conference on Learning Representations*, 2025.

Wenqian Ye, Guangtao Zheng, Xu Cao, Yunsheng Ma, Xia Hu, and Aidong Zhang. Spurious correlations in machine learning: A survey. *arXiv preprint arXiv:2402.12715*, 2024.

Sriram Yenamandra, Pratik Ramesh, Viraj Prabhu, and Judy Hoffman. Facts: First amplify correlations and then slice to discover bias. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 4794–4804, 2023.

Jingzhao Zhang, Aditya Menon, Andreas Veit, Srinadh Bhojanapalli, Sanjiv Kumar, and Suvrit Sra. Coping with label shift via distributionally robust optimisation. *International Conference on Learning Representations*, 2021.

Hongke Zhao, Chuang Zhao, Xi Zhang, Nanlin Liu, Hengshu Zhu, Qi Liu, and Hui Xiong. An ensemble learning approach with gradient resampling for class-imbalance problems. *INFORMS Journal on Computing*, 35(4):747–763, 2023.

Ziliang Samuel Zhong, Xiang Pan, and Qi Lei. Bridging domains with approximately shared features. *arXiv preprint arXiv:2403.06424*, 2024.

A Proof of Proposition 3.2.1

Proof of Proposition 3.2.1. In a single tree, any three samples have a unique lowest common ancestor structure. Let $u = \mathrm{LCA}(\boldsymbol{x}_i, \boldsymbol{x}_j)$ and $v = \mathrm{LCA}(\boldsymbol{x}_j, \boldsymbol{x}_k)$. By the tree hierarchy, the LCA of $(\boldsymbol{x}_i, \boldsymbol{x}_k)$ occurs at least as high as the shallower of u and v, which implies

$$\mathrm{Depth}_{\mathrm{LCA}}(\boldsymbol{x}_i, \boldsymbol{x}_k) \geq \min(\mathrm{Depth}_{\mathrm{LCA}}(\boldsymbol{x}_i, \boldsymbol{x}_j), \mathrm{Depth}_{\mathrm{LCA}}(\boldsymbol{x}_j, \boldsymbol{x}_k)).$$

This is equivalent to the proposition.

B Implementation Details

For SMOTE-based methods, the number of nearest neighbors is set to $k = 5$ for all datasets. For TVAE, CTGAN, CTGAN-ENN, and DGOT, we adopt the default hyperparameters provided in their official implementations (Xiao et al., 2021; Adiputra & Wanchai, 2024; Ren et al., 2025). For ensemble methods, we use the default hyperparameter settings provided by the benchmark library (Liu, 2025).

TreeSMOTE constructs a Random Forest with 100 trees to capture the hierarchical structure of the data. Each tree is grown using Gini impurity as the split criterion, with no depth restriction, and each leaf node contains at least one sample. At each split, a random subset of features is considered when searching for the best split, following the standard feature subsampling strategy used in Random Forests, where the number of candidate features is set to $\sqrt{d}$ for classification tasks, with d denoting the total number of features. When combined with an undersampling ensemble method, TreeSMOTE performs class-wise data augmentation on each balanced subset, doubling the number of samples in each class.

Each experiment is repeated five times with different random seeds, and the average results are reported. For the downstream classifiers, Classification Tree, LinearSVC, and MLP use the default hyperparameters provided by the scikit-learn package, while LightGBM uses the default settings from its official implementation. For each dataset, samples are randomly split into 75% training and 25% testing before applying any resampling or augmentation. Nominal variables are encoded into one-hot vectors for fair comparison among different methods, as some of them are not capable of processing nominal variables without dummification.

B.1 Summary statistics of datasets

We present summary statistics of all datasets used in our experiments, including the CLIMB benchmark (grouped by imbalance level) and the datasets from the imbalanced-learn package.

Table 4: Summary statistics of the CLIMB benchmark datasets, grouped by imbalance level.

Imbalance level	Statistic	samples	features	IR
Low (28 datasets)	min	161	3	2.01
	max	175,028	107	3.95
	mean	22,562.21	24.5	2.67
Medium (24 datasets)	min	327	4	5.17
	max	101,766	117	9.92
	mean	11,885.71	37.92	7.3
High (15 datasets)	min	294	4	10
	max	149,639	85	42.01
	mean	35,001.8	19.91	27.4
Extreme (6 datasets)	min	3,772	9	64.03
	max	284,807	36	577.88
	mean	56,229.33	23.83	201.82

Table 5: Summary statistics of datasets from the *imbalanced-learn* package.

Dataset	samples	features	IR
optical_digits	5,620	64	9.14
satimage	6,435	36	9.28
pen_digits	10,992	16	9.42
sick_euthyroid	3,163	42	9.8
thyroid_sick	3,772	52	15.33
coil_2000	9,822	85	15.76
wine_quality	4,898	11	25.77
letter_img	20,000	16	26.25
abalone_19	4,177	10	129.53
isolet	7,797	617	11.99

B.2 Metrics

The evaluation metrics used in this study are defined as follows:

- Recall: $\frac{TP}{TP+FN}$, where TP is the number of true positives and FN is the number of false negatives. It measures the ability of the model to correctly identify positive instances.
- Precision: $\frac{TP}{TP+FP}$, where FP is the number of false positives. It measures the accuracy of the positive predictions made by the model.
- AUPRC: The area under the precision-recall curve, which emphasizes the precision-recall trade-off for the positive (typically minority) class, making it particularly informative in imbalanced settings.

- BAC: $\frac{1}{2}\left(\frac{TP}{TP+FN}+\frac{TN}{TN+FP}\right)$, where TN is the number of true negatives. Balanced Accuracy assigns equal weight to the recall of each class, preventing dominance by the majority class.
- F1-Score: $2 \cdot \frac{Precision \cdot Recall}{Precision+Recall}$, which is the harmonic mean of precision and recall, providing a single metric that balances both.
- G-mean: $\sqrt{Precision \cdot Recall}$, which is the geometric mean of precision and recall, providing a single metric that balances both, and is particularly useful in imbalanced datasets as it penalizes low values in either precision or recall.
- MCC: $\frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}$, which considers all four entries of the confusion matrix and therefore provides a more balanced evaluation for imbalanced classification problems.

C results

C.1 Results of SMOTE-Based Methods

Table 6: Comparison of TreeSMOTE with classical oversampling methods across different imbalance levels (low, medium, high, extreme) on AUPRC.

Classifier	Oversampling Method	Imbalance Level				
		Low	**Medium**	**High**	**Extreme**	**Average**
DecisionTree	IDENTITY	50.49±2.36	50.90±2.65	34.02±2.58	42.90±5.76	46.62±2.78
	SMOTE	50.86±2.39	50.09±2.27	33.35±2.26	34.08±4.16	45.63±2.47
	BorderlineSMOTE	50.07±2.53	50.12±2.47	33.24±1.94	37.10±4.36	45.56±2.54
	SVMSMOTE	50.49±2.48	50.73±2.79	33.78±1.94	38.55±3.43	46.15±2.55
	SMOTE-TomekLinks	50.94±2.61	50.79±2.61	32.73±2.39	34.91±3.21	45.83±2.61
	TreeSMOTE	51.26±2.35	51.71±2.60	34.47±2.00	42.89±5.31	47.27±2.60
LinearSVC	IDENTITY	68.56±2.99	66.33±2.06	42.10±2.54	54.06±3.24	61.20±2.61
	SMOTE	68.30±3.07	65.18±2.42	40.51±2.87	52.53±3.32	60.27±2.84
	BorderlineSMOTE	67.94±2.97	64.33±2.43	37.93±2.75	52.44±3.75	59.31±2.81
	SVMSMOTE	68.06±2.83	65.28±2.25	39.62±2.86	53.20±3.77	60.08±2.72
	SMOTE-TomekLinks	68.35±3.07	65.65±2.16	40.95±2.87	52.13±3.58	60.50±2.77
	TreeSMOTE	68.60±2.97	65.59±2.32	41.13±2.78	53.63±3.37	60.73±2.75
MLP	IDENTITY	71.49±2.60	70.68±2.10	52.46±2.35	63.02±4.11	66.62±2.51
	SMOTE	71.03±2.79	69.62±2.22	51.28±2.69	61.19±3.61	65.70±2.65
	BorderlineSMOTE	69.87±2.64	69.41±2.07	51.27±2.68	61.81±3.82	65.23±2.56
	SVMSMOTE	70.42±2.49	69.94±2.16	52.21±2.56	62.09±4.18	65.84±2.54
	SMOTE-TomekLinks	70.91±2.90	69.64±2.12	51.15±2.82	61.58±3.70	65.67±2.69
	TreeSMOTE	71.14±2.49	70.18±2.12	52.03±2.28	63.75±4.58	66.29±2.50
LightGBM	IDENTITY	72.35±2.26	73.31±2.05	57.67±3.03	54.79±6.89	68.35±2.72
	SMOTE	72.17±2.33	73.12±1.99	57.93±2.59	65.94±5.00	69.20±2.49
	BorderlineSMOTE	72.00±2.11	72.81±1.94	58.11±2.15	60.38±6.58	68.60±2.43
	SVMSMOTE	72.38±2.61	73.16±1.91	58.42±2.44	62.85±6.49	69.13±2.67
	SMOTE-TomekLinks	71.85±2.07	73.27±2.03	58.17±2.00	61.61±6.04	68.81±2.37
	TreeSMOTE	72.37±2.33	73.39±1.97	58.74±2.31	66.65±7.65	69.58±2.65

Table 7: Comparison of TreeSMOTE with classical oversampling methods across different imbalance levels (low, medium, high, extreme) on BAC.

Classifier	**Oversampling Method**	**Imbalance Level**				
		Low	**Medium**	**High**	**Extreme**	**Average**
DecisionTree	IDENTITY	71.71±1.96	74.45±2.09	60.86±1.97	75.19±3.33	70.67±2.12
	SMOTE	72.71±2.11	75.35±1.85	63.02±2.31	75.61±4.57	71.83±2.27
	BorderlineSMOTE	72.23±2.20	75.24±2.18	61.95±1.82	73.81±3.60	71.24±2.23
	SVMSMOTE	72.39±2.23	75.74±2.12	62.74±1.82	74.65±2.80	71.70±2.16
	SMOTE-TomekLinks	72.97±2.15	75.44±1.98	62.07±2.13	75.54±2.85	71.75±2.15
	TreeSMOTE	72.63±2.10	75.57±1.82	61.85±1.33	75.63±4.56	71.63±2.05
LinearSVC	IDENTITY	71.67±1.67	70.16±1.31	51.43±1.34	62.98±1.71	66.30±1.49
	SMOTE	76.27±2.26	78.87±1.74	66.09±1.96	82.32±3.77	75.53±2.15
	BorderlineSMOTE	75.96±2.24	78.24±1.71	65.09±1.71	80.71±3.48	74.87±2.06
	SVMSMOTE	76.01±2.24	78.34±1.74	64.41±1.89	81.01±3.68	74.80±2.12
	SMOTE-TomekLinks	76.39±2.38	76.41±1.74	62.56±1.64	74.31±3.82	73.38±2.13
	TreeSMOTE	76.24±2.18	78.20±1.57	62.83±1.62	79.34±3.35	74.39±1.96
MLP	IDENTITY	75.07±1.73	75.76±1.43	61.88±1.93	69.95±3.41	72.17±1.81
	SMOTE	77.82±2.14	78.96±1.62	67.80±1.95	78.81±3.61	76.22±2.05
	BorderlineSMOTE	77.61±1.87	78.99±1.67	67.25±1.70	78.06±3.83	75.97±1.93
	SVMSMOTE	77.77±2.00	79.16±1.67	67.37±1.79	77.59±4.22	76.08±2.03
	SMOTE-TomekLinks	77.85±2.10	77.87±1.66	65.61±2.03	75.27±3.63	75.13±2.07
	TreeSMOTE	77.61±1.78	78.66±1.60	66.68±2.02	77.77±3.57	75.72±1.92
LightGBM	IDENTITY	75.32±1.63	76.87±1.78	65.64±2.35	69.51±5.38	73.47±2.13
	SMOTE	76.97±2.07	78.77±1.55	70.58±1.97	78.40±3.51	76.45±2.00
	BorderlineSMOTE	77.24±2.13	78.62±1.52	70.34±1.91	78.17±4.62	76.44±2.09
	SVMSMOTE	77.11±2.02	79.08±1.47	69.91±2.02	77.31±3.26	76.38±1.94
	SMOTE-TomekLinks	77.24±2.02	78.48±2.01	68.78±1.93	75.03±3.58	75.82±2.13
	TreeSMOTE	76.58±2.01	78.49±1.61	69.15±1.87	77.78±3.62	75.87±1.99

Table 8: Comparison of TreeSMOTE with classical oversampling methods across different imbalance levels (low, medium, high, extreme) on F1-Score.

Classifier	**Oversampling Method**	**Imbalance Level**				
		Low	**Medium**	**High**	**Extreme**	**Average**
Classification Tree	None	71.65±1.95	74.23±1.92	60.76±1.76	75.30±3.03	70.56±1.99
	SMOTE	72.17±2.01	74.09±1.57	60.57±1.74	70.19±3.46	70.26±1.93
	BorderlineSMOTE	71.60±2.09	74.16±1.89	60.61±1.45	71.80±2.97	70.20±1.96
	SVMSMOTE	71.85±2.17	74.42±2.08	61.08±1.49	72.48±2.13	70.53±2.00
	SMOTE-TomekLinks	72.40±2.11	74.31±1.88	60.28±1.78	71.16±2.36	70.44±1.99
	TreeSMOTE	72.36±2.04	74.81±1.81	61.40±1.21	74.77±3.08	71.11±1.88
LinearSVC	None	71.62±1.68	71.15±1.51	52.52±1.35	66.38±1.30	67.11±1.52
	SMOTE	73.68±2.18	73.00±1.53	52.62±1.20	57.11±1.37	67.77±1.70
	BorderlineSMOTE	72.41±2.14	71.46±1.67	52.50±1.37	60.04±1.97	66.99±1.81
	SVMSMOTE	73.20±2.29	73.14±1.66	53.66±1.82	62.67±2.07	68.30±1.97
	SMOTE-TomekLinks	73.77±2.28	71.58±1.55	50.84±1.12	56.80±1.30	66.94±1.72
	TreeSMOTE	74.78±2.12	74.80±1.49	55.71±1.49	66.95±1.90	70.23±1.77
MLP	None	75.03±1.83	76.59±1.48	63.00±1.93	72.30±3.24	72.85±1.85
	SMOTE	75.99±2.09	77.01±1.51	62.90±1.73	73.46±2.02	73.43±1.82
	BorderlineSMOTE	75.20±1.89	76.91±1.61	63.54±1.64	74.44±2.48	73.30±1.80
	SVMSMOTE	75.87±2.08	77.56±1.60	64.36±1.83	74.20±2.69	73.93±1.92
	SMOTE-TomekLinks	76.05±2.12	76.46±1.50	62.16±1.78	73.10±2.47	73.09±1.88
	TreeSMOTE	76.80±1.84	77.77±1.44	64.84±1.80	75.59±2.36	74.56±1.74
LightGBM	None	75.90±1.72	78.04±1.73	67.06±2.57	71.83±4.59	74.55±2.13
	SMOTE	76.93±2.03	78.86±1.39	68.38±1.86	76.28±2.51	75.86±1.82
	BorderlineSMOTE	76.86±2.08	78.71±1.34	68.81±1.73	76.96±3.89	75.92±1.92
	SVMSMOTE	76.79±2.03	79.06±1.38	68.99±1.97	77.02±2.96	76.05±1.88
	SMOTE-TomekLinks	77.13±2.00	78.88±1.92	67.86±1.80	73.19±2.74	75.59±1.99
	TreeSMOTE	76.71±2.00	79.06±1.50	69.46±1.83	79.11±2.81	76.28±1.87

Table 9: Comparison of TreeSMOTE with classical oversampling methods across different imbalance levels (low, medium, high, extreme) on G-mean.

Classifier	**Oversampling Method**	**Imbalance Level**				
		Low	**Medium**	**High**	**Extreme**	**Average**
DecisionTree	IDENTITY	71.71±1.94	74.31±1.90	60.80±1.76	75.53±3.08	70.64±1.99
	SMOTE	72.27±2.02	74.22±1.59	60.92±1.77	70.90±3.50	70.46±1.95
	BorderlineSMOTE	71.68±2.11	74.26±1.91	60.79±1.44	72.12±3.06	70.33±1.98
	SVMSMOTE	71.93±2.18	74.52±2.07	61.26±1.50	72.85±2.20	70.66±2.00
	SMOTE-TomekLinks	72.50±2.10	74.46±1.89	60.54±1.78	71.82±2.32	70.63±1.98
	TreeSMOTE	72.42±2.05	74.89±1.80	61.46±1.20	75.09±3.16	71.20±1.89
LinearSVC	IDENTITY	72.57±1.75	72.38±1.56	53.68±1.38	67.13±1.17	68.18±1.56
	SMOTE	74.30±2.15	74.20±1.48	55.57±1.25	61.46±1.35	69.36±1.68
	BorderlineSMOTE	73.33±2.11	72.82±1.57	55.25±1.33	63.61±1.83	68.65±1.75
	SVMSMOTE	73.83±2.24	73.96±1.63	55.84±1.67	65.69±1.91	69.51±1.90
	SMOTE-TomekLinks	74.40±2.27	73.06±1.55	53.59±1.15	60.36±1.48	68.53±1.74
	TreeSMOTE	75.10±2.08	75.32±1.43	56.98±1.49	68.79±1.95	70.93±1.74
MLP	IDENTITY	75.62±1.88	77.11±1.52	63.83±2.05	72.99±3.30	73.47±1.91
	SMOTE	76.41±2.08	77.33±1.49	63.82±1.69	74.21±2.09	73.94±1.81
	BorderlineSMOTE	75.74±1.84	77.24±1.57	64.18±1.58	74.91±2.50	73.79±1.75
	SVMSMOTE	76.19±2.02	77.74±1.57	64.81±1.76	74.60±2.74	74.23±1.88
	SMOTE-TomekLinks	76.46±2.10	76.89±1.49	63.14±1.78	73.69±2.50	73.64±1.87
	TreeSMOTE	76.99±1.78	77.91±1.42	65.12±1.83	75.85±2.44	74.76±1.72
LightGBM	IDENTITY	76.28±1.78	78.47±1.70	68.00±2.59	72.93±4.53	75.12±2.14
	SMOTE	77.03±2.03	79.06±1.37	68.84±1.86	76.96±2.44	76.11±1.81
	BorderlineSMOTE	76.97±2.08	78.93±1.32	69.18±1.75	77.47±3.82	76.15±1.91
	SVMSMOTE	76.90±2.03	79.34±1.36	69.34±2.01	77.52±2.93	76.29±1.88
	SMOTE-TomekLinks	77.24±2.01	79.21±1.84	68.46±1.83	73.95±2.79	75.91±1.98
	TreeSMOTE	76.82±2.00	79.30±1.56	69.85±1.87	79.93±2.72	76.55±1.89

Table 10: Comparison of TreeSMOTE with classical oversampling methods across different imbalance levels (low, medium, high, extreme) on MCC.

Classifier	**Oversampling Method**	**Imbalance Level**				
		Low	**Medium**	**High**	**Extreme**	**Average**
DecisionTree	IDENTITY	43.52±3.88	56.04±3.20	41.92±3.15	56.17±5.03	48.35±3.60
	SMOTE	44.69±4.06	55.74±2.80	41.52±3.05	49.55±4.37	48.07±3.46
	BorderlineSMOTE	43.51±4.25	56.07±3.29	41.37±2.26	51.71±4.50	47.87±3.54
	SVMSMOTE	43.98±4.37	56.39±3.57	42.30±2.47	52.97±4.34	48.45±3.71
	SMOTE-TomekLinks	45.16±4.18	56.88±3.11	41.45±2.88	50.28±3.06	48.67±3.47
	TreeSMOTE	44.96±4.11	57.19±2.99	42.58±2.47	57.47±4.76	49.52±3.46
LinearSVC	IDENTITY	47.12±3.72	55.31±2.65	33.32±2.27	45.71±2.51	46.86±2.97
	SMOTE	49.74±4.26	57.37±2.55	35.94±2.08	34.53±2.40	48.16±3.10
	BorderlineSMOTE	48.38±4.18	55.16±2.43	34.73±2.42	37.39±3.21	46.90±3.16
	SVMSMOTE	48.80±4.40	56.21±2.64	35.61±2.99	40.56±3.22	47.85±3.44
	SMOTE-TomekLinks	49.96±4.52	57.76±2.54	36.03±1.95	37.06±3.06	48.60±3.22
	TreeSMOTE	50.76±4.11	58.37±2.41	37.15±2.59	45.37±2.97	50.02±3.14
MLP	IDENTITY	52.58±3.72	61.92±2.42	47.29±3.25	54.21±4.39	54.70±3.25
	SMOTE	53.60±4.19	61.65±2.41	46.57±2.60	55.72±3.49	54.98±3.22
	BorderlineSMOTE	52.48±3.73	61.48±2.55	47.53±2.49	56.83±4.55	54.78±3.15
	SVMSMOTE	52.95±3.94	62.18±2.59	48.28±2.81	56.96±4.42	55.35±3.30
	SMOTE-TomekLinks	53.66±4.23	61.66±2.40	46.82±2.62	56.11±3.45	55.09±3.23
	TreeSMOTE	54.32±3.51	62.59±2.32	48.43±2.93	58.47±3.82	56.17±3.03
LightGBM	IDENTITY	53.31±3.61	63.45±2.81	50.81±3.77	52.47±5.78	56.14±3.56
	SMOTE	54.21±4.07	64.48±2.46	51.25±2.71	60.60±3.05	57.59±3.18
	BorderlineSMOTE	54.13±4.14	64.13±2.34	51.80±2.75	61.88±5.03	57.66±3.34
	SVMSMOTE	53.96±4.08	65.02±2.57	51.82±2.84	62.95±3.76	57.98±3.31
	SMOTE-TomekLinks	54.63±4.04	65.00±2.99	51.16±2.87	58.00±4.83	57.69±3.53
	TreeSMOTE	53.81±3.99	64.88±2.82	52.83±2.81	65.79±4.32	58.31±3.40

C.2 Results of Generative Model-Based Methods

Table 11: AUPRC comparison of TreeSMOTE and generative model-based oversampling methods on 10 datasets.

Dataset	TVAE	CTGAN	CTGANENN	DGOT	TreeSMOTE
optical_digits	83.46±0.82	74.01±1.63	86.76±1.93	89.08±1.49	90.08±1.60
satimage	36.54±1.88	36.27±1.98	34.67±1.30	39.30±1.95	41.37±2.25
pen_digits	88.95±0.88	87.41±0.78	86.69±2.50	92.55±0.98	92.81±0.81
sick_euthyroid	65.48±3.82	46.33±4.27	52.59±13.41	77.58±2.73	77.81±2.73
thyroid_sick	61.70±5.77	44.27±5.63	52.15±10.45	76.77±6.04	79.55±2.91
coil_2000	11.90±0.68	10.47±1.14	10.11±0.79	12.34±0.64	12.10±0.58
wine_quality	26.14±4.60	18.55±2.55	17.32±3.44	26.18±4.93	26.86±5.65
letter_img	80.94±4.93	81.19±1.96	81.69±1.33	90.80±0.78	89.62±1.15
abalone_19	2.19±1.08	3.29±2.00	2.29±0.86	1.96±0.24	4.39±3.12
isolet	77.74±1.54	79.61±1.00	78.22±1.24	77.91±1.09	81.37±1.20

Table 12: BAC comparison of TreeSMOTE and generative model-based oversampling methods on 10 datasets.

Dataset	TVAE	CTGAN	CTGANENN	DGOT	TreeSMOTE
optical_digits	92.04±1.11	90.56±0.68	93.20±0.98	92.47±0.96	94.61±1.14
satimage	70.53±1.37	64.56±0.66	60.72±1.78	63.84±2.99	74.81±1.59
pen_digits	96.51±0.21	95.83±0.29	92.90±0.52	94.84±0.31	96.93±0.50
sick_euthyroid	89.39±1.38	78.40±2.70	69.71±4.98	84.50±2.27	92.03±1.26
thyroid_sick	86.93±1.64	76.77±2.60	67.98±4.53	79.48±3.26	91.31±1.06
coil_2000	53.88±0.80	53.01±1.62	52.14±0.69	52.21±0.44	56.22±0.76
wine_quality	66.78±1.57	69.01±2.23	59.71±2.29	57.43±1.59	67.21±2.12
letter_img	93.88±1.23	95.74±0.33	94.92±0.60	92.68±0.58	96.89±0.55
abalone_19	56.24±6.15	54.95±3.23	52.77±2.04	50.19±0.61	51.08±1.84
isolet	89.76±0.52	89.43±0.54	83.40±0.90	87.58±0.68	90.58±0.63

Table 13: F1-Score comparison among minority data augmentation by a generative model and TreeSMOTE.

Dataset	TVAE	CTGAN	CTGAN-ENN	DGOT	TreeSMOTE
optical_digits	87.19±0.76	82.33±0.98	90.14±1.23	91.15±1.06	91.85±1.21
satimage	55.23±1.37	52.03±0.96	49.77±1.55	52.22±2.33	59.10±1.50
pen_digits	90.66±0.51	90.34±0.32	89.35±0.98	93.82±0.85	94.19±0.52
sick_euthyroid	76.20±3.31	64.14±3.38	63.58±8.21	81.94±1.86	84.09±1.82
thyroid_sick	73.35±2.72	62.15±3.52	62.59±6.63	76.96±4.54	83.59±1.27
coil_2000	36.56±0.83	34.85±1.72	33.51±0.71	35.03±0.47	38.05±0.61
wine_quality	51.23±2.68	45.60±2.28	43.76±2.59	44.42±2.45	51.43±3.34
letter_img	84.62±2.31	85.37±1.56	86.71±0.79	92.26±0.57	91.60±0.60
abalone_19	31.82±2.45	32.55±2.00	31.52±0.86	30.47±0.46	32.43±3.61
isolet	83.96±0.92	85.07±0.62	81.85±0.65	83.64±0.79	86.45±0.84

Table 14: G-mean comparison of TreeSMOTE and generative model-based oversampling methods on 10 datasets.

Dataset	TVAE	CTGAN	CTGANENN	DGOT	TreeSMOTE
optical_digits	90.13±0.52	86.80±0.73	92.69±0.86	93.56±0.79	93.66±0.85
satimage	67.05±0.97	64.72±0.80	63.22±1.25	64.23±1.98	69.45±1.27
pen_digits	91.98±0.42	92.14±0.23	91.80±0.61	95.43±0.76	95.33±0.34
sick_euthyroid	81.60±2.92	74.02±2.52	74.02±5.73	86.98±1.67	87.70±1.50
thyroid_sick	79.65±1.67	72.47±2.53	73.47±4.91	82.21±3.66	86.85±0.69
coil_2000	54.27±0.70	52.67±1.55	50.95±0.55	52.56±0.40	55.51±0.46
wine_quality	65.87±2.02	60.06±1.90	60.53±1.87	59.78±1.69	65.85±2.51
letter_img	87.21±1.48	87.66±1.44	89.37±0.64	94.48±0.43	92.98±0.39
abalone_19	49.55±1.28	50.95±1.47	50.63±0.55	49.98±0.47	51.76±3.50
isolet	88.09±0.67	89.09±0.43	86.96±0.51	88.18±0.56	90.08±0.64

Table 15: MCC comparison of TreeSMOTE and generative model-based oversampling methods on 10 datasets.

Dataset	TVAE	CTGAN	CTGANENN	DGOT	TreeSMOTE
optical_digits	80.34±1.03	74.05±1.40	85.41±1.71	87.14±1.58	87.38±1.71
satimage	36.24±2.03	30.59±1.76	28.24±2.18	30.89±4.04	42.45±1.84
pen_digits	84.51±0.80	84.66±0.39	83.66±1.16	90.88±1.53	90.77±0.67
sick_euthyroid	64.19±5.34	48.61±4.95	48.40±11.46	74.16±3.24	75.76±2.89
thyroid_sick	59.82±3.26	45.67±4.78	47.39±9.79	65.00±7.08	74.04±1.39
coil_2000	8.81±1.28	6.14±3.14	3.62±1.17	6.15±0.82	11.44±0.91
wine_quality	31.87±4.03	22.66±3.46	21.22±3.79	20.47±3.42	31.89±4.94
letter_img	74.83±2.79	75.76±2.72	79.16±1.23	88.98±0.86	86.18±0.78
abalone_19	3.21±3.31	3.61±2.82	1.79±1.08	0.26±0.94	3.64±7.04
isolet	76.25±1.34	78.20±0.86	74.03±1.03	76.38±1.12	80.17±1.27

C.3 Results of Ensemble-Based Methods

Table 16: Performance comparison between ensemble methods and their combinations with TreeSMOTE across different imbalance levels (low, medium, high, and extreme) on AUPRC.

Method	Low	Medium	High	Extreme	Average
SelfPacedEnsemble	71.55±2.44	71.65±1.82	54.66±1.86	64.86±4.82	67.39±2.31
SelfPacedEnsemble w/ TreeSmote	71.67±2.62	72.16±1.70	54.68±1.49	65.07±4.57	67.63±2.23
BalancedRandomForest	71.40±3.01	70.46±1.89	51.89±2.61	59.09±5.44	65.84±2.75
BalancedRandomForest w/ TreeSmote	71.04±2.89	70.23±2.26	51.97±2.46	58.95±5.44	65.64±2.80
RUSBoost	71.09±2.96	69.31±2.31	49.85±2.38	56.05±5.66	64.64±2.84
RUSBoost w/ TreeSmote	70.79±2.60	69.98±2.42	50.54±2.49	56.54±5.39	64.95±2.75
UnderBagging	70.94±2.73	70.50±2.04	52.96±2.34	59.35±6.26	65.94±2.71
UnderBagging w/ TreeSmote	71.03±2.91	70.33±2.23	52.70±2.39	58.40±5.20	65.78±2.76
BalanceCascade	70.69±2.59	70.55±2.03	51.03±2.35	63.00±4.69	65.77±2.53
BalanceCascade w/ TreeSmote	70.88±2.92	70.49±2.13	50.99±2.16	63.00±4.66	65.81±2.63

Table 17: Performance comparison between ensemble methods and their combinations with TreeSMOTE across different imbalance levels (low, medium, high, and extreme) on BAC.

Method	Low	Medium	High	Extreme	Average
SelfPacedEnsemble	77.21±2.09	80.83±1.79	69.66±1.74	85.03±2.85	77.51±1.98
SelfPacedEnsemble w/ TreeSmote	77.11±2.30	80.97±1.67	69.92±1.67	85.04±3.13	77.57±2.02
BalancedRandomForest	78.38±2.15	82.22±1.41	71.33±1.57	86.17±2.61	78.85±1.81
BalancedRandomForest w/ TreeSmote	78.19±2.21	82.34±1.48	71.51±1.41	86.35±2.96	78.88±1.85
RUSBoost	77.00±2.39	80.29±1.43	68.61±1.75	84.16±3.33	76.94±2.00
RUSBoost w/ TreeSmote	77.35±2.27	80.81±1.58	69.53±1.96	83.80±3.67	77.41±2.09
UnderBagging	77.64±2.28	81.75±1.69	71.19±1.32	85.96±2.54	78.38±1.89
UnderBagging w/ TreeSmote	77.60±2.18	81.75±1.52	71.45±1.80	86.40±2.59	78.46±1.91
BalanceCascade	76.56±2.19	80.27±1.57	66.48±2.05	83.23±3.53	76.24±2.06
BalanceCascade w/ TreeSmote	76.96±2.23	80.05±1.72	66.35±1.70	83.60±2.82	76.31±1.99

Table 18: Performance comparison between ensemble methods and their combinations with TreeSMOTE across different imbalance levels (low, medium, high, and extreme) on F1-Score.

Method	**Low IRs**	**Medium IRs**	**High IRs**	**Extreme IRs**	**Average**
SelfPacedEnsemble	76.88±1.99	78.84±1.36	64.61±1.18	73.93±2.12	74.67±1.61
SelfPacedEnsemble w/ TreeSmote	76.73±2.09	78.98±1.30	64.78±1.21	74.31±2.20	74.74±1.64
BalancedRandomForest	75.85±2.11	75.84±1.37	58.87±1.23	55.49±1.85	70.46±1.64
BalancedRandomForest w/ TreeSmote	75.71±2.11	76.07±1.37	59.45±1.13	55.80±1.04	70.64±1.55
RUSBoost	75.80±2.31	76.55±1.31	60.45±1.35	62.28±1.73	71.61±1.71
RUSBoost w/ TreeSmote	76.14±2.24	77.03±1.49	60.90±1.58	62.25±2.08	71.99±1.83
UnderBagging	76.50±2.13	77.46±1.55	61.67±1.09	58.71±1.64	72.12±1.67
UnderBagging w/ TreeSmote	76.44±2.10	77.50±1.39	62.04±1.36	59.17±1.18	72.24±1.62
BalanceCascade	76.32±2.02	78.29±1.36	60.22±1.43	69.29±2.59	72.94±1.72
BalanceCascade w/ TreeSmote	76.74±2.16	77.97±1.39	60.33±1.40	70.05±2.38	73.07±1.75

Table 19: Performance comparison between ensemble methods and their combinations with TreeSMOTE across different imbalance levels (low, medium, high, and extreme) on G-mean. .

Method	**Low**	**Medium**	**High**	**Extreme**	**Average**
SelfPacedEnsemble	77.01±1.99	79.23±1.40	65.68±1.22	75.74±1.88	75.24±1.61
SelfPacedEnsemble w/ TreeSmote	76.88±2.11	79.37±1.31	65.88±1.22	76.10±2.01	75.31±1.63
BalancedRandomForest	76.39±2.08	77.13±1.32	61.43±1.25	60.78±1.58	72.10±1.60
BalancedRandomForest w/ TreeSmote	76.26±2.08	77.34±1.36	61.88±1.06	61.07±0.94	72.25±1.52
RUSBoost	76.01±2.33	77.13±1.30	61.91±1.38	65.81±1.75	72.50±1.72
RUSBoost w/ TreeSmote	76.34±2.24	77.61±1.48	62.46±1.58	65.70±1.92	72.89±1.81
UnderBagging	76.68±2.14	78.11±1.55	63.35±1.01	63.36±1.48	73.17±1.64
UnderBagging w/ TreeSmote	76.62±2.11	78.17±1.38	63.70±1.33	63.82±1.10	73.29±1.61
BalanceCascade	76.39±2.04	78.64±1.36	61.37±1.46	71.73±2.40	73.54±1.71
BalanceCascade w/ TreeSmote	76.80±2.15	78.34±1.44	61.47±1.37	72.32±2.12	73.66±1.74

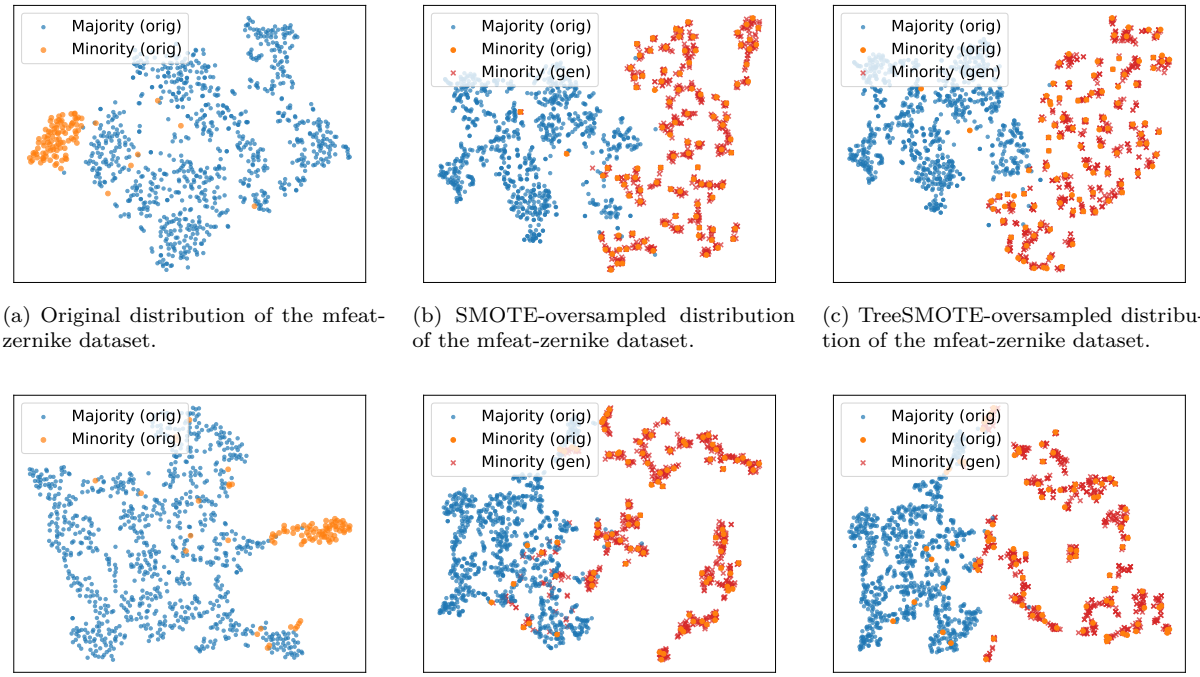
Table 20: Performance comparison between ensemble methods and their combinations with TreeSMOTE across different imbalance levels (low, medium, high, and extreme) on MCC.

Method	**Low**	**Medium**	**High**	**Extreme**	**Average**
SelfPacedEnsemble	54.26±3.99	64.84±2.46	50.29±2.22	59.09±3.42	57.45±3.04
SelfPacedEnsemble w/ TreeSmote	54.00±4.25	65.13±2.30	50.79±2.16	59.35±3.58	57.59±3.07
BalancedRandomForest	53.73±4.14	61.73±2.16	43.75±1.96	33.53±2.27	52.60±2.83
BalancedRandomForest w/ TreeSmote	53.50±4.14	62.30±2.33	44.58±1.55	33.98±1.96	52.93±2.78
RUSBoost	52.39±4.71	60.89±2.19	44.26±2.40	40.09±2.66	52.51±3.17
RUSBoost w/ TreeSmote	53.00±4.47	61.85±2.55	44.88±2.51	39.51±2.98	53.14±3.26
UnderBagging	53.68±4.30	63.01±2.62	45.76±1.74	36.39±2.46	53.70±3.02
UnderBagging w/ TreeSmote	53.56±4.24	63.07±2.22	46.32±2.20	37.28±1.86	53.87±2.91
BalanceCascade	52.89±4.11	63.67±2.29	42.42±2.40	50.59±4.12	54.14±3.12
BalanceCascade w/ TreeSmote	53.72±4.29	63.25±2.57	42.38±2.31	51.11±3.43	54.33±3.20

C.4 Visualization

To provide an intuitive view of how TreeSMOTE affects the data distribution for real data, we visualize the sample distributions using t-SNE. Specifically, we select two datasets, *mfeat-zernike* and *Pulsar-Dataset-HTRU2*, and compare the original data distribution with those obtained after applying SMOTE and TreeSMOTE. For each dataset, we project the high-dimensional samples onto a two-dimensional space using t-SNE.

Figure 3 visualizes the results. For each dataset, we present three cases: the original data distribution, the distribution after applying SMOTE, and the distribution after applying TreeSMOTE. Compared with SMOTE, TreeSMOTE generates samples with less noise in the feature space, incurring class boundaries with a larger margin.



(a) Original distribution of the mfeat-zernike dataset.

(b) SMOTE-oversampled distribution of the mfeat-zernike dataset.

(c) TreeSMOTE-oversampled distribution of the mfeat-zernike dataset.

(d) Original distribution of the Pulsar-Dataset-HTRU2 dataset.

(e) SMOTE-oversampled distribution of the Pulsar-Dataset-HTRU2 dataset.

(f) TreeSMOTE-oversampled distribution of the Pulsar-Dataset-HTRU2 dataset.

Figure 3: t-SNE visualizations of data distributions on two datasets. For the *mfeat-zernike* dataset (Fig. 3a–3c), the original distribution, SMOTE, and TreeSMOTE results are presented. Compared with SMOTE, TreeSMOTE produces class boundaries with a larger margin. For the *Pulsar-Dataset-HTRU2* dataset (Fig. 3d–3f), the original data and the distributions after SMOTE and TreeSMOTE are shown. SMOTE tends to introduce noisy samples that may distort the data distribution, whereas TreeSMOTE better preserves the underlying data structure.

C.5 Ablation Study

We conduct an ablation study to analyze the contributions of different components in our method. Specifically, we investigate from two perspectives: (1) the proposed structural distance versus Euclidean distance for finding nearest neighbors, and (2) using versus not using consistency filtering for synthetic sample elimination.

Table 21 presents the results of LightGBM as the downstream classifier across five evaluation metrics on extremely imbalanced datasets. The results show that using the proposed structural distance achieves an

Table 21: Ablation study of TreeSMOTE with LightGBM as the classifier on extremely imbalanced data.

Distance	Filtering	AUPRC	BAC	F1-Score	G-mean	MCC
Euclidean	No	65.94	78.40	76.28	76.96	60.60
Euclidean	Yes	65.98	78.29	78.98	79.6	65.69
Structural	No	66.26	**78.69**	76.61	77.26	61.38
Structural	Yes	**66.65**	77.78	**79.11**	**79.93**	**65.79**

overall better performance than the Euclidean distance. Furthermore, applying consistency filtering improves results by excluding potentially noisy samples, thereby increasing the reliability of the generated synthetic data.