International Journal of Mathematical, Engineering and Management Sciences
Vol. 11, No. 4, 1792-1829, 2026
https://doi.org/10.33889/IJMEMS.2026.11.4.073

KFCGAN-OVO: A Novel Kalman Filter-Integrated CGAN Approach for Imbalanced Multiclass Classification

Satyendra Singh Rawat
Department of Computer Science and Engineering,
Amity University, Gwalior, Madhya Pradesh, India.
Corresponding author: satyandra.rawat@s.amity.edu, satyandra2005@gmail.com

Vikas Thada
Department of Computer Science and Engineering,
Amity University, Gwalior, Madhya Pradesh, India.
E-mail: vthada@gwa.amity.edu

Amit Kumar Mishra
School of Computer Science and Engineering,
Chandigarh University Uttar Pradesh, Unnao, Uttar Pradesh, India.
E-mail: amitkmishra.phd@gmail.com

(Received on December 29, 2025; Revised on April 14, 2026 & June 4, 2026 & June 17, 2026; Accepted on June 20, 2026)

Abstract
Multiclass imbalanced classification is much harder than binary classification because of the major challenges with data sets, including high dimensionality, class overlap, noise, small disjuncts, and a small number of minority class samples. The oversampling technique is largely used to address class imbalance issues. Several oversampling approaches such as SMOTE, ADASYN, Borderline-SMOTE, etc., heavily rely on the k-nearest neighbors (KNN) algorithm, which classifies samples based on the majority class of their selected k-nearest neighbors. Additionally, the Euclidean distance measure imposes further restrictions on the classification process. Due to these concerns, we propose a new hybrid approach called KFCGAN-OVO, which is specifically made for multiclass imbalanced problems, and it does not rely on SMOTE, k-nearest neighbors' selection, or Euclidean distance. The proposed approach employs a Kalman filter for noise reduction and data smoothing; a Conditional Generative Adversarial Network (CGAN), a generative model that generates realistic synthetic data based on given conditions or labels, for dataset balancing; and a one-vs-one (OVO) data decomposition technique to manage high dimensionality. KFCGAN-OVO's effectiveness is analyzed on several measures of data complexity, performance, and computational costs. The results indicate that it outperformed its competitors on all measures.

Keywords- Multiclass imbalance, Data complexity, Generative adversarial network, Kalman filter, OVO decomposition.

1. Introduction

Most attempts to date have merely addressed the imbalance between the two classes. One of two types of multiclass can be seen in an imbalanced dataset: multimajority cases, which have a minority and several majority classes, or multiminority cases, which have a majority and several minority classes. Such problems may involve numerous minority classes and majority classes. Concerns of multiclass imbalance persist in real-world applications (Wang & Yao, 2012). Learning classifier ensembles using rebalanced data is a popular method. Both undersampling and oversampling of the majority class and minority class samples, respectively, are examples of bootstrap averaging, or bagging. Rebalancing techniques, however, may introduce their biases by making unequal changes to the examples of various classes. Moreover, these methodologies often necessitate a priori (pre-learning) delineation of the pertinent performance metric (Wang & Awang, 2025). One of the main issues that many popular Machine learning algorithms face is class imbalance, which is defined as an imbalance in the frequency of classes. This difficulty prompted the

creation of numerous strategies to address the disparity between classes (Altalhan et al., 2025; Cemernek et al., 2024). Compared to binary imbalance problems, multiclass imbalance problems are more difficult to learn and have fewer study findings. To solve issues with data imbalance, resampling techniques are frequently used. Current resampling techniques, primarily designed for binary imbalance datasets, exhibit notable limitations when applied to multiclass imbalance datasets (Wang & Awang, 2024). Traditional classification methods tend to underrepresent minority classes while favoring majority classes, which may be significant in a number of applications. The existence of label noise makes this classification problem much worse by making it more difficult to determine the best decision borders between classes and possibly causing model overfitting (Brishti et al., 2025). Machine learning (ML) techniques are increasingly being used in various sectors, including disease prediction (Ganie & Pramanik, 2024; Kannan et al., 2025), fraud detection (Martínez et al., 2025), anomaly detection (Shana et al., 2025), fault diagnosis (Geetha & Geethanjali, 2024), etc., due to their capacity to identify solutions among the intricate interactions of variables. Nevertheless, the size and distribution of the data used to train prediction models have an impact on them. Due to the rarity of class interest, class imbalance is a prevalent problem in these applications (Bae et al., 2025).

The Synthetic Minority Oversampling Technique (SMOTE) (Chawla et al., 2002) and Generative Adversarial Network (GAN) (Goodfellow et al., 2014; Mirza & Osindero, 2014) are two of the most popular approaches to address class imbalance issues in data sets. However, the limitations of the current methods have prompted the development of novel variants to address these problems (Udu et al., 2025). The SMOTE preprocessing method is considered the “de facto” standard when learning from imbalanced data. Several SMOTE variants have developed since its inception, but in the current scenario, it is incapable of dealing with small disjuncts, noise, data scarcity, overlaps, data shifts, and the issue of dimensionality in the multiclass data sets (Fernández et al., 2018). The majority of synthetic oversampling techniques create synthetic instances inside the convex hull formed by the existing minority instances because they focus on the minority class and disregard the wealth of information offered by the majority class. Additionally, they frequently perform poorly on highly imbalanced data because there are fewer minority cases, which means there is less material to create synthetic instances from (Khorshidi & Aickelin, 2025). In artificial intelligence (AI), generative adversarial networks (GANs) are one of the most researched topics. Their exceptional ability to generate data has drawn a lot of interest (Pan et al., 2019). Artificial intelligence is seeing an increase in real-world applications, but it also brings with it several hidden problems (Cemernek et al., 2024). Class imbalance constantly plagues machine learning, as datasets often lack sufficient samples for minority classes. Current solutions frequently create artificial data to address this problem, but they usually have trouble handling complex data distributions, mainly because they prioritize oversampling the minority class while ignoring its interactions with the majority class (Wang et al., 2025). A high degree of similarity can raise the risk of re-identification; however, synthetic data is intended to preserve statistical similarities to the original data while protecting sensitive information (Min & Oh, 2025). To address the issues of SMOTE, the authors introduced a hybrid SMOTE-AAE resampling method. In this method, SMOTE first generates artificial samples for the minority class. Then, an Adversarial Autoencoder (AAE) refines these samples to balance the datasets. The results indicate that this SMOTE-AAE works better than other methods, especially for large imbalanced binary datasets (Rawat et al., 2026).

The structure of the paper is as follows:
Section 1: Provide the paper's introduction.
Section 2: This section contains related work.
Section 3: This section discusses materials and methods.
Section 4: This section contains the proposed work.
Section 5: The different performance metrics for multiclass problems are covered in this section.

Section 6: This section presents the findings and discussions.
Section 7: This section provides the conclusion and recommendations for the future.

2. Related Work

Based on Mahalanobis distance (Ghorbani, 2019; Xie & Huang, 2024), a novel oversampling methodology known as MDO (Mahalanobis Distance-based Oversampling technique) generates synthetic samples that are equivalent in distance from the considered class mean to other minority class cases. It seeks to resolve the novel problems that multiclass skewed distributions provide. MDO more effectively addresses the class overlapping issues, a major problem in multiclass imbalance scenarios. Its performance is assessed on small datasets and only supports numerical attributes (Abdi & Hashemi, 2016). A novel Synthetic Minority Oversampling Method (SMOM) is proposed based on k-nearest neighbor selection to address multiclass imbalance issues. SMOM supports only continuous features when working with imbalanced data sets with nominal and ordinal attributes (Zhu et al., 2017). PT-bagging was suggested as a simple plug-in approach for imbalanced classification. The approach has two limitations: (1) it does not examine suitability in dynamic cost contexts; and (2) it ignores intrinsic data properties for handling difficult scenarios, including small disjuncts (Collell et al., 2018). The difficulty of the multiclass classification can be reduced by decomposing the original multiclass problem into binary class problems by using the One-vs-One (OVO) decomposition strategy. The authors introduce a novel approach, Distance-based Relative Competence Weighting with Adaptive Synthetic Example Generation (DRCW-ASEG), that successfully tackles the imbalance issue. It is inappropriate for handling large imbalanced multiclass problems (Zhang et al., 2018). The performance of the classification can be more affected by learning from the imbalanced data with other complex issues such as noise, overlapping class distribution, and small disjuncts. Most research in the area of imbalance learning focuses on binary classification problems; the more critical multiclass problems remain mostly unresolved. This paper presents the novel oversampling technique called Multiclass Combined Cleaning and Resampling (MC-CCR), a data pre-processing technique for imbalanced multiclass data sets that combines instance cleaning and resampling methods to improve class distribution and reduce noise prior to the model training. Unlike SMOTE, MC-CCR models the region-wise oversampling using an energy-based approach that is less affected by small disjuncts and outliers. The majority of instances that are close to the minority instances are not addressed (Koziarski et al., 2020). The use of several techniques to address data imbalance, particularly neighborhood-based oversampling algorithms based on SMOTE, may be restricted by data difficulty factors such as small disjuncts, the existence of outliers, and a lack of training observations. Radial-Based Oversampling (RBO) (Koziarski et al., 2019) can mitigate the drawbacks of neighborhood-based approaches. Based on the idea of mutual class potential, this work suggested a novel undersampling algorithm called Radial-Based Undersampling (RBU). The RBU method is a better option than the radial-based oversampling algorithm since it is easier to understand and uses less computing power. When the dataset has a limited number of rare and outlier minority cases, RBU performance goes down (Koziarski, 2020). Multiclass imbalance and concept drift have made a number of real-world applications more difficult. This paper presents Class-Aware Labeling for Multi-class Imbalanced Data (CALMID), a complete active learning method for streaming data with multiple classes that are not evenly distributed (Liu et al., 2021). The standard approach to address problems with multiclass imbalance is to use data resampling, which either oversamples instances of the minority class or undersamples instances of the majority class. Oversampling can cause an overfitting problem, but undersampling can mean losing some critical information. The authors suggest a new method called Cluster-based Hybrid Sampling for Imbalanced Data (CBHSID) to fix these issues. One of the problems with the recommended method is that it wasn't tested on noisy data (Palli et al., 2022). The dynamic nature of real-world systems is a significant challenge for deployed predictive machine learning (ML) models. Changes made to the system that trained the machine learning model over time could make the system work less well. Recent advancements in the examination of dynamic environments have predominantly focused

on identifying and addressing the transitions induced by a phenomenon termed concept drift (Bayram et al., 2022).

The crucial area of research in online supervised learning is rapidly evolving data streams, including manufacturing, health, and the environment. The class imbalance that frequently exists in these sectors may affect class distributions. Researchers don't accord enough consideration to multiclass imbalanced situations with different levels of imbalance in evolving streams. To bridge this knowledge gap, this study introduces the Dynamic Queues (DynaQ) algorithm for online learning in multiclass imbalanced environments. The algorithm does not address extremely skewed distributions, where minority examples may appear in spurts (Sadeghi et al., 2023). This paper focuses on the complex issues of concept drift, fluctuating multiclass imbalance, a constrained label cost budget, and one-by-one processing. Although the MicFoal architecture for classifying online network traffic proposed in this paper is based on active learning, it does not yet have a concept drift detector for situations involving multiclass imbalance (Liu et al., 2023). Complex data characteristics found in real-world data for multiclass classification problems frequently result in lower classification results. A heterogeneous feature space that increases the number and complexity of class patterns and a high degree of multiclass imbalance within the data present significant analytical hurdles. Current classification or data pre-processing methods only address one of these two issues separately. In their innovative classification approach, the authors specifically address the issues of heterogeneous feature space and multiclass imbalance. The suggested approach has the limitation of requiring a hierarchical taxonomic structure to create homogeneous sample subsets (Hirsch et al., 2023). The imbalance ratio, along with other factors that make data harder, like class overlap or breaking up a minority class into numerous sub-concepts, has a big effect on how well the classification works. This paper presents GMMSampling, an innovative resampling method that concurrently oversamples each subconcept of the minority class while eliminating class-overlapping regions from majority class instances, leveraging insights on data difficulty characteristics. It uses Gaussian Mixture Models (GMM) to get close to the distribution of the minority class. However, fitting a GMM becomes much more expensive as the data gets more complicated because its models only work with numbers (Naglik & Lango, 2024). Oversampling is a common technique for class balancing that advances the training data's generalization. More naive methods have the potential to insert additional biases into the data, and they are particularly vulnerable to errors in the training data—a challenge given the typically noisy data collected in the healthcare industry. Particularly with high-dimensional data, this is an issue. To enhance other less sophisticated generative methods, a generative adversarial network-based technique is suggested for producing synthetic samples from small, high-dimensional data (Cusworth et al., 2024). Multilabel classification research mainly concentrates on learning from each instance that can be associated with multiple classes or labels. It can be challenging to adapt single-label approaches to multilabel learning to address the class-imbalance problem because many of their core concepts are hard to transfer. This paper proposes to use evolutionary computation to oversample the minority class and undersample the majority class for multilabel problems. This method has the hidden effect of reducing the datasets' size because of its undersampling capability (García-Pedrajas et al., 2024). This paper proposes an HSSADL-CAC method for dealing with data that is not evenly distributed across classes, together with a perfect DL model for finding cyberattacks. It has problems with scalability and real-time performance, especially in big or changing contexts (Alabduallah et al., 2024). To solve the problem of multiclass imbalance in identifying internet traffic with noise and overlaps, a hybrid method that used membership-density oversampling was suggested. The method may not work well when dealing with datasets that are high-dimensional and unbalanced (Hartono & Syah, 2024). This paper proposes a semi-supervised variational bi-directional sampling (SVBDS) technique exclusively applicable in the electric power sector to mitigate the effects of imbalanced data on accurate electric power failure identification (Qin et al., 2024). The MKC-SMOTE approach successfully resolves multiclass imbalance problems and significantly enhances classification performance. However, the

method may face challenges, including determining a suitable value for k in k-nearest neighbors and handling the extremely small number of minority class samples (Wang & Awang, 2024). Localized Ensemble Learning (LEL) was described as a principled approach that simultaneously addresses the three fundamental imbalances of quality, distance, and neighborhood by adapting sampling and learning to safe, borderline, and rare minority circumstances. There are still limits to computational complexity for big databases (Olabisi et al., 2025). To address class imbalance in tabular data, this study introduces Conditional Variational Autoencoders supplemented with contrastive learning (CTVAE), a novel method. However, it is only applicable in situations involving tabular data (Wang et al., 2025). The GDHS approach broadens the generalization of minority samples and methodically handles overlapping regions to enhance classification performance for multiclass imbalanced datasets. While an adaptive number of k is selected based on the dataset, a fixed value of k (i.e., k=5) is utilized for the k-nearest neighbors' selection (Yan et al., 2025). Multiclass imbalanced datasets can be efficiently classified using the OCH-SMOTE algorithm. However, the technique may encounter difficulties with datasets that have a significant degree of class imbalance, small classes (with fewer than five samples each), and class overlap. Because it uses the OVO decomposition approach, this algorithm may be computationally costly (Wang & Awang, 2025). Both binary and multiclass imbalance problems are successfully handled by SOMM, which can efficiently produce a variety of synthetic cases inside the minority data space for actual data sets. SOMM is useful at many levels, especially when handling a limited number of minority instances. The capacity of SOMM to produce synthetic instances is constrained when the majority class is dispersed throughout the data space and the spaces of the majority and minority classes significantly overlap (Khorshidi & Aickelin, 2025). This study introduces the novel oversampling algorithm SMOTE-CLS, which is dependent on the advantages of SMOTE and VAE. Compared to conventional oversampling methods, this merger makes training more complex. Its suitability for multiclass tasks is questioned since it can be difficult to modify the latent space according to the confusion matrix in such a scenario (Hong et al., 2025).

Table 1 lists the advantages and limitations of the various techniques for dealing with imbalanced multiclass problems. To address the challenges and concerns mentioned in this section, we define our research objectives (ROs) as follows:

RO1: Several methods for handling class imbalances rely on Euclidean distance and k-nearest neighbor selection, both of which have notable limitations (Suwanda et al., 2020; Zhang, 2021). How can the challenges of imbalanced multiclass problems be effectively addressed without depending on these approaches?

RO2: Compared to binary-class imbalanced datasets, multiclass imbalanced datasets are considerably more challenging to handle. How can we efficiently classify multiclass problems that are high-dimensional, exhibit class overlap, or suffer from severe class imbalance?

RO3: The difficulty of multiclass problems is generally high, and quantifying this difficulty across all complexity measures is challenging. How do balancing techniques perform when confronted with the challenges posed by such problem difficulty, particularly in terms of data complexity?

Table 1. Compare some of the most recent approaches to address multiclass imbalance issues.

Reference	Method	Technology used	Advantages	Limitations
Abdi & Hashemi (2016)	MDO	Friedman test, KNN, C4.5, Mahalanobis distance, ROS, SMOTE, ADASYN	Reduce the risk of overlapping between different classes.	It does not handle nominal and mixed-type(numeric/nominal) attributes.
Zhu et al. (2017)	SMOM	k-NN, NBDOS, two-round filtration	To deal with multiclass imbalanced problems.	It only suitable to handle continuous attributes.
Naglik & Lango (2024)	GMMSampling	Data difficulty factors	To deal with multiclass imbalance data.	It is unable to capture intricate situations where minority classes overlap significantly.
Guan et al. (2024)	MFCC-GAN	Mel-Frequency Cepstral Coefficient (MFCC), GAN	To address the problems of class imbalance and small sample sizes in pipeline fault diagnosis.	The data are collected only from the simulation platform of the oil and gas pipeline network.
Alabduallah et al. (2024)	HSSADL-CAC	Hybrid Salp Swarm Algorithm (HSSA), Deep Extreme Learning Machine (DELM) model, ADASYN	To address class imbalance in data handling for cyberattack detection.	Limited scalability and run-time performance in dynamic or large-scale scenarios.
Olabisi et al. (2025)	Localised ensemble learning (LEL)	KNN, Random Forest, Decision Tree, XGBoost, and Naive Bayes.	A localized approach to address class imbalance problem.	The computational complexity of large datasets still has limitations.
Asmat et al. (2025)	Spectra Balance GAN (SB-GAN)	Conditional GAN (CGAN)	Reduce majority class bias and Improve Classification for Hyperspectral Imaging (HSI).	Deep learning-based models—particularly Conditional GAN (CGAN) models—take longer to run.
Ramachandra & Vaithiyanathan (2025)	Fed-DPSDG-WGAN	Min-Max, WGAN, Autoencoders, CTGAN	Improve loan defaulter prediction by creating synthetic data.	For large-scale privacy-preserving synthetic data generation, efficiency and scalability are constrained.
Khorshidi & Aickelin (2025)	Synthetic Oversampling with Minority and Majority classes (SOMM)	Min-Max Normalization, MDO, SMOM, GMMSampling, HCE-MCD	Both binary and multiclass imbalance problems can be addressed with low time complexity.	There is substantial overlap between the spaces of the majority and minority classes.
Wang & Awang (2025)	One-Class Hypersphere SMOTE (OCH-SMOTE)	One-vs-One (OVO) decomposition, CH-SMOTE	To address multiclass imbalanced datasets.	Computationally expensive. Datasets with extremely small classes (less than five samples per class) could be difficult for it to handle.
Hong et al. (2025)	SMOTE-Class-Level Sampling (SMOTE-CLS)	SMOTE, Variational Autoencoder (VAE)	It was robust to noise and greatly enhanced the minority class's classification performance.	It is only applicable to multi-class tasks and has a higher computational cost than traditional methods.
Wang et al. (2025)	CTVAE	Variational Autoencoders (VAE) Contrastive learning techniques.	Addresses the class imbalance issues in tabular data.	It is inapplicable to datasets with intricate structures and a variety of domains.

3. Materials and Methods

3.1 Generative Adversarial Network

Classic Generative Adversarial Networks (GANs) (Creswell et al., 2018) are expanded upon by Conditional Generative Adversarial Networks (CGANs) (Mert, 2023), which give both the discriminator and generator additional information, such as class labels or data attributes. Because of this conditioning, the model can generate data that is consistent with predefined standards and realistic. While the generator learns to produce data samples conditioned on this input, the discriminator evaluates the original data's authenticity and conformance to the condition. This technique is particularly effective for applications such as image-to-image translation, text-to-image synthesis, and data augmentation. Let

$x \sim P_{data}$ (x) be a real data sample.

$z \sim P_z$(z) be a latent noise vector.

$y \epsilon \mathcal{Y}$ be the conditional variable (e.g., class label).

The generator learns a mapping: $G: (z, y) \rightarrow x' = G\ (z, y)$, where x' is a synthetic sample conditioned by y. The discriminator evaluates whether a sample is real or fake given by y, $D: (x, y) \rightarrow [0, 1]$, where D (x, y) presents the probability that x is real under condition y.

The following is the updated objective function:

$$min_G\ max_D V\ (D, G) = E_{x \sim P_{data}} [\log D\ (x|y)] + E_{z \sim P_z} [\log(1 - D(G(z|y)|y))] \quad (1)$$

In the fundamental design, the discriminator D and generator G regularly play a minimax game when an input, y (such as a class label or image), is added. More control over the data generation process is possible with this formulation than with traditional GANs (Mirza & Osindero, 2014). **Figure 1** shows the fundamental CGAN design. A sufficient amount of data is usually required to properly train deep learning (DL) classifiers without an overfitting issue. In such cases, generative networks may be used when a massive amount of data is required. The data collection process is problematic due to several constraints. To enhance the accuracy of the DL classifier in situations where the sample size is inappropriate, CGAN can generate synthetic samples of a particular category. The proposed method integrates a CNN classifier with a CGAN to enhance accuracy through data augmentation (Wickramaratne & Mahmud, 2021). In reference to network intrusion detection, this research examines the problem of anomaly detection in imbalanced situations in datasets. This introduces a novel approach to anomaly detection that addresses the problem of class imbalance by applying both data-level and algorithm-level methods. This approach combines the auto-learning capabilities of reinforcement learning with the oversampling capability of a CGAN. The paper also investigates the effects of CGAN-based oversampling on the classifiers LR, RF, MLP, and NB to determine how well a CGAN can perform on imbalanced classification problems. Experimental findings show that the proposed method using CGAN-based oversampling in general outperforms current oversampling techniques, such as ADASYN and synthetic minority oversampling (Ahsan et al., 2022). The drawbacks of traditional Intrusion Detection Systems (IDS) in handling irregular data and the problem of missing attack categories are handled by a proposed intrusion detection technique using CGAN. The attack instances produced by the CGAN are arbitrarily distributed within a predetermined range and closely resemble the pattern of the input data to prevent data from mechanical expansion from being repeated. CGAN makes attack instances that roughly match the distribution pattern of the input data and are randomly spread out within a set range to avoid the repetition that comes from mechanically expanding the data. The experiments show that this method can lower the number of missed detections caused by unbalanced data classes and amounts, while also raising the accuracy of classification. Also, its overall performance has a greater ability to generalize and superior performance metrics (Zhao et al., 2023). The uses based on Generative Adversarial Networks (GANs) are increasing in popularity, which has shown in-depth study and analysis in the different fields. Natural language processing, text-to-image, image-to-image, audio-to-image, 3D object creation, prediction, and architectural design are all uses for GAN models. For both production and prediction, this approach is essential to preserving visual integrity and security. In the case of facial recognition, it is very advantageous for identifying fake images. GANs are vital in examining visual legitimacy on social media by identifying and assessing frauds. As research advances, new GAN variations appear, which prompts the creation of various analysis techniques to predict the models' applicability and efficacy. This paper presents detail about the several recent developments to GAN model designs, the pros and cons of different types of GANs, possible solutions to their problems, and the new domains where GAN tools could be useful in the future. Furthermore, it examines critical measures like Inception Score (IS) and Fréchet Inception Distance (FID) as essential benchmarks for improving GAN performance relative to existing methods (Sharma et al., 2024).

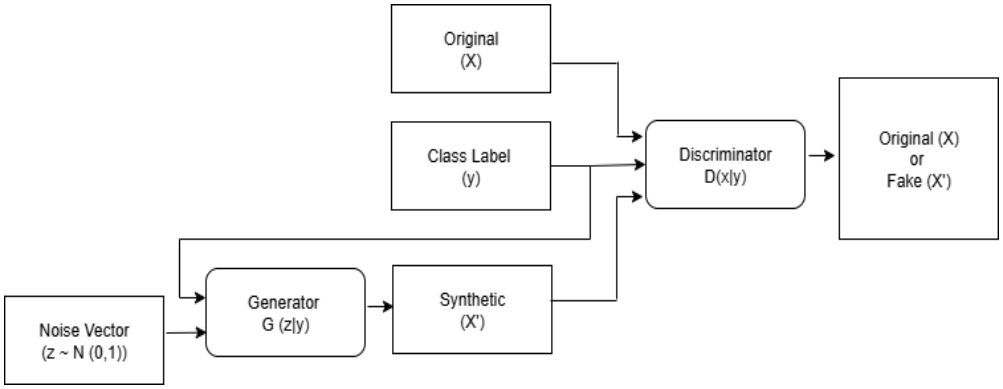


Figure 1. Conditional generative adversarial network.

3.2 Feature-wise Sequential (Scalar) Kalman Filter for Temporal Smoothing

The scalar Kalman Filter (KF) (Kalman, 1960) for noise reduction in multiclass problems denoises sequential or per-sample features independently before feeding them into multiclass classifiers (e.g., SVM, Random Forest), treating each feature dimension as a univariate linear Gaussian process to minimize mean squared error and sharpen class separability.

For dataset $X \in \mathbb{R}^{N \times d}$ (N samples, d features) with additive noise (Kalman, 1960)

$$z_{t,i} = x_{t,i} + v_{t,i} \; (v_{t,i} \sim \mathcal{N}(0, R_i)) \tag{2}$$

per feature i =1…d at time t =1…N (or per-sample), model state evolution as constant-velocity or constant.

$$x_{t,i} = F_i x_{t-1,i} + w_{t,i} \; (w_{t,i} \sim \mathcal{N}(0, Q_i)) \tag{3}$$

observed as

$$z_{t,i} = H_i x_{t,i} + v_{t,i} \tag{4}$$

Typically, $F_i = 1$ (random walk smoothing), $H_i = 1$ (scalar case). Here,

F_i: State transition scalar (e.g., 1 for smoothing/random walk).
H_i: Observation scalar (often 1 for direct features).
Q_i: Process noise variance ($w_{t,i} \sim \mathcal{N}(0, Q_i)$); models dynamics uncertainty.
R_i: Measurement noise variance ($v_{t,i} \sim \mathcal{N}(0, R_i)$).
$v_t \sim \mathcal{N}(0, R)$: Zero-mean Gaussian measurement noise.
$x_{t,i}$: True underlying state (clean feature value) for feature i at t; scalar ∈ $\mathbb{R}$.
$z_{t,i}$: Noisy observation/measurement of $x_{t,i}$:

$$z_{t,i} = H_i x_{t,i} + v_{t,i} \tag{5}$$

Prediction (per feature i, all t):

$$\hat{x}_{t|t-1,i} = F_i \hat{x}_{t-1|t-1,i}, P_{t|t-1,i} = F_i^2 P_{t-1|t-1,i} + Q_i \tag{6}$$

where, $\hat{x}_{t|t-1,i}$ is the predicted state estimate before measurement, $\hat{x}_{t|t-1,i} = F_i \hat{x}_{t-1|t-1,i}$, $\hat{x}_{t|t,i}$ is the updated posterior state estimate after measurement (denoised output), and $P_{t|t-1,i}$: is the predicted error variance (uncertainty):

$$P_{t|t-1,i} = F_i^2 P_{t-1|t-1,i} + Q_i > 0 \tag{7}$$

Update with noisy input $z_{t,i}$:

$$K_{t,i} = \frac{P_{t|t-1,i}H_i}{H_i^2 P_{t|t-1,i} + R_i} \tag{8}$$

$$\hat{x}_{t|t,i} = \hat{x}_{t|t-1,i} + K_{t,i}(z_{t,i} - H_i\hat{x}_{t|t-1,i}) \tag{9}$$

$$P_{t|t,i} = (1 - K_{t,i}H_i)P_{t|t-1,i} \tag{10}$$

where, $K_{t,i}$ is the Scalar Kalman gain, $P_{t|t,i}$ is the posterior error variance. Output denoised matrix $X' = [\hat{x}_{t|t,i}]_{t,i}$; initialize $\hat{x}_{1|1,i} = z_{1,i}$, $P_{1|1,i} = R_i$ (Haykin, 2001; Ma et al., 2020).

Algorithm 1 presents a sequential Kalman filter that is applied to every feature across samples. It sequentially estimates and updates each feature's value to reduce measurement noise, producing a smoothed feature representation X_s.

Algorithm 1: Kalman Smoothing of Features.

Input:

$X \in \mathbb{R}^{n\times d}$: Input dataset with n samples and d features.

Q: Process variance (default 1×10^{-4}).

R: Measurement variance (default 1×10^{-1}).

Output:

$X_s \in \mathbb{R}^{n\times d}$: Smoothed feature matrix.

Procedure:

I. Initialize a zero matrix X_s of size $n \times d$.

II. *for* each feature $j = 1,2,\ldots,d$:

a. Set initial state estimate $\hat{x}_0 = X_{1,j}$.

b. Initialize error covariance $P = 1.0$.

c. *for* each sample $i = 1,2,\ldots,n$:

i. Observation $z = X_{i,j}$.

ii. *Prediction step:*

- Predicted state: $\hat{x}_i^- = \hat{x}_{i-1}$
- Predicted covariance: $P^- = P + Q$ (11)

iii. *Update step:*

- Kalman gain: $K = \frac{P^-}{P^- + R}$ (12)
- Updated state estimate: $\hat{x}_i = \hat{x}_i^- + K(z - \hat{x}_i^-)$ (13)
- Updated covariance: $P = (1 - K)P^-$ (14)

iv. Assign $X_s[i,j] = \hat{x}_i$.

d. *end for*

III. *end for*

IV. Return the smoothed dataset X_s.

3.3 One-versus-One Data Decomposition

In the One-versus-One (OVO) data decomposition scheme, a n-class problem is divided into $n(n\text{-}1)/2$ binary subproblems. Each subproblem is faced by independent base classifiers, which are responsible for distinguishing the instances from the different pairs of classes. With such appropriate consideration, the much more complex multiclass problem is translated into the simpler binary class subproblems, which is

expected to receive better results or address multiclass problems with binary classification techniques (Zhang et al., 2016). In the prediction phase, a test pattern is classified according to the outputs of each binary classifier, organized as the following score-matrix R:

$$R = \begin{bmatrix} - & r_{12} & \cdots & r_{1n} \\ r_{21} & - & \cdots & r_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ r_{n1} & \cdots & \cdots & - \end{bmatrix} \quad (15)$$

where, $r_{ij} \in [0, 1]$ is the confidence of the binary classifier distinguishing class i from j, whereas, the confidence in favor of class j is computed as $r_{ji} = 1 - r_{ij}$ (if it is not provided by the classifier). Once the score-matrix is produced, any aggregation methods (Galar et al., 2011) can be employed to infer the final class. The weighted voting strategy has proven to be one of the best aggregation methods due to its robustness and effectiveness (Galar et al., 2011; Hüllermeier & Vanderlooy, 2010). In weighted voting, the final output class is computed over the score-matrix R as shown in the following expression:

$$\text{class} = \arg max_{i=1,\ldots,n} \sum_{j=1, j\neq i}^{n} s_{ij}, \text{ where } s_{ij} = \begin{cases} 1 & if\ r_{ij} > r_{ji} \\ 0 & Otherwise \end{cases} \quad (16)$$

3.4 Data Complexity

We can enhance a better understanding of the characteristics that influence the difficulty of learning activities by using data complexity analysis (Wan et al., 2023). The goal of complexity measurements is to show how hard it is to work with supervised data. These measures deal with problems like overlap, density, linearity, and so on that affect how well ML classifiers work (Lancho et al., 2023). One of the main issues researchers face when using machine learning on data is the varying difficulty of predictions (Kwon et al., 2024). This research assesses the classifier's performance from the view of data complexity. The data complexity metrics are divided into six classes: neighborhood (N1, N2, N3, N4, T1, LSC), network (Density, ClsCoef, Hubs), dimensionality (T2, T3, T4), linearity (L1, L2, L3), balance (C1, C2), and overlap (F1, F1.v, F2, F3, F4) (Eberlein et al., 2025).

4. Proposed Work

4.1 System Configurations and Parameter-tuning

We have employed the following system configurations:

- Google Colab: Python 3 Google Compute Engine backend (Tensor Processing Unit), 47.0 GB system RAM, 225.3 GB Disk.
- PyTorch, TensorFlow, and Imblearn Python libraries (Pedregosa et al., 2011).
- Problexity Python library (Komorniczak & Ksieniewicz, 2022).
- The laptop is equipped with a 500 GB hard disk, a 13th Gen Intel® Core™ i5-1335U Processor, 8.0 GB installed RAM, and an Intel® Iris® Xe Graphics card.

Table 2 provides the parameter tuning for the methods.

Table 2. The methods' parameter tuning.

Methods	Parameter settings
CGAN	epochs=200, batch_size = 64, verbose = False, noise_dim = 32, device = 'cpu', hidden_dim = 128, learning_rate= 2e-4, LeakyRelu (0.2), adam , BCE Loss, PyTorch.
Kalman Filter	Process_var = 1e-4, meas_var = 1e-4
Stratified K-Folds Cross-Validation	K= 10, shuffle = True, random_state = 42
Decision Tree	random_state = 42, criteria = "gini"
SVM	Kernel = 'rbf', probability = True, random_state = 42

4.2 Datasets

We use the multiclass datasets listed below (see **Table 3**), sourced from the UCI repository. For ease of use, we have given every dataset a distinct identity. The dataset's identity (ID), name, number of instances (N), features (F), classes (C), class-wise imbalance ratio (Class-wise IR), class-wise distribution of instances (Class-Distribution), and overall imbalance ratio (Overall IR) are the columns that make up **Table 3**. The overall imbalance ratio is used for arranging all datasets in ascending order. The datasets include both mixed (numerical and categorical) and/or numerical features. For each of the imbalanced multiclass datasets, we compute the imbalance ratios across classes (overall IR) and within classes (class-wise IR). The ratio of majority class instances to minority class instances is known as the imbalance ratio. A dataset is considered imbalanced if the imbalance ratio is more than one.

Table 3. Description of the imbalanced multiclass datasets used in our experiments.

ID	Dataset	N	F	C	Class distribution	Class-wise IR	Overall IR
win	Wine	178	13	03	59/71/48	1.20/1.00/1.47	1.47
cmc	Contraceptive Method Choice	1473	09	03	629/333/511	1.00/1.89/1.23	1.89
bal	Balance Scale	625	4	03	49/288/288	5.87/1.00/1.00	5.87
dry	Dry Bean	13611	16	07	1322/522/1630/3546/1928/2027/2636	2.68/6.79/2.17/1.00/1.84/1.75/1.34	6.79
gla	Glass Identification	214	09	06	70/76/17/13/9/29	1.09/1.00/4.47/5.85/8.44/2.62	8.44
zoo	Zoo	101	16	07	41/20/5/13/4/8/10	1.00/2.05/8.20/3.15/10.25/5.12/4.10	10.25
car	Car Evaluation	1728	06	04	384/69/1210/65	3.15/17.53/1.00/18.61	18.61
lym	Lymphography	148	19	04	2/81/61/4	40.50/1.00/1.32/20.25	40.50
wqr	Wine Quality-Red	1599	12	06	10/53/681/638/199/18	68.10/12.84/1.00/1.06/3.42/37.84	68.10
yes	Yeast	1484	8	10	463/429/244/163/51/44/35/30/20/5	1.00/1.07/1.90/2.84/9.08/10.52/13.23/15.43/23.15/92.60/	92.60
abs	Absenteeism at work	740	19		44/88/157/112/60/7/1/208/19/16/6/7/1/2/3/3/1/2/3/	4.72/2.36/1.32/1.85/3.46/29.71/208.00/1.00/10.94/13.00/34.67/29.71/208.00/104.00/69.34/69.34/208.00/104.00/69.34	208.00
wqw	Wine Quality-White	4898	12	07	20/163/1457/2198/880/175/5	109.90/13.48/1.50/1.00/2.49/12.56/439.60	439.60
aba	Abalone	4177	08	28	1/1/15/57/115/259/391/568/689/634/487/267/203/126/103/67/58/42/32/26/14/6/9/2/1/1/2/1	689.00/689.00/45.93/12.08/5.99/2.66/1.76/1.21/1.00/1.08/1.41/2.58/3.39/5.46/6.68/10.28/11.87/16.40/21.53/26.50/49.21/114.83/76.55/344.50/689.00/689.00/344.50/689.00	689.00
nur	Nursery	12960	08	05	4320/4266/2/4044/328	1.00/ 1.01/2160.00 / 1.06/13.17	2160.00

4.3 KFCGAN Method

The hybrid KFCGAN resampling method uses a scalar Kalman Filter and a CGAN to do sequential feature denoising and class-conditional data creation, respectively. This makes it possible to do robust multiclass classification of noisy data.

The following steps are performed in this sequence:

Step 1. Scalar Kalman Filter (Feature Denoising): Apply KF independently per feature $i = 1 \dots d$ on input $X \in \mathbb{R}^{N \times d}$ using Equations (5) to (8) and yields denoised $X' = [\hat{x}_{t|t,i}]$. Typically, $F_i = H_i = 1$.

Step 2. CGAN Data Augmentation (Class-Conditional Generation):

(a) Generate synthetic samples per class $c = 1 \dots C$ using Equation (1).

(b) Generator: $G(\mathrm{z}, c) \rightarrow \hat{x}_c$ (noise z, class label c.

(c) Discriminator: $D(\mathrm{x}, c) \rightarrow [0,1]$ (real/fake conditioned on c).

Augment X' with $\hat{X}_c$ per class, balancing imbalanced datasets.

Algorithm 2: KFCGAN

Input:

$\mathcal{D} = \{(x_i, y_i)\}$: Multiclass dataset with Kclasses

CGAN: Conditional GAN architecture (Generator G, Discriminator D, and training parameters)

target_strategy: Strategy for balancing minority classes.

Output: Balanced training dataset $\mathcal{D}_{bal}$

Procedure:

I. Preprocessing:
   a. Normalize or standardize feature columns in $\mathcal{D}$.
   b. Split $\mathcal{D}$ into training and testing subsets: $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{test}}$.

II. Feature Smoothing:
   Apply Algorithm 1 (Feature-wise Scalar Kalman Filter) on $\mathcal{D}_{\text{train}}$ to obtain smoothed features $\mathcal{D}'_{\text{train}}$.

III. Class Distribution Computation:
   For each class $c \in \{1,2,\ldots,K\}$:
   a. Compute sample count $n_c = |\{x_i : y_i = c\}|$.
   b. Determine target count $n_{\text{target}}[\mathrm{c}]$ using target_strategy.

IV. Conditional GAN Training:
   a. Initialize Generator G and Discriminator D based on CGAN.
   b. For each training epoch:
   i. For each minibatch $(x_{real}, y_{real}) \in \mathcal{D}'_{train}$:
   - Sample latent noise $z \sim p(z)$.
   - Generate synthetic samples $x_{fake} = \mathrm{G}(z, y_{real})$.
   - Update discriminator D to maximize:

$$\mathcal{L}_D = \mathbb{E}[log\, D(x_{real}, y_{real})] + \mathbb{E}[log(1 - D(x_{fake}, y_{real}))] \quad (17)$$

   - Update generator G to minimize:

$$\mathcal{L}_G = \mathbb{E}[log(1 - D(G(z, y_{real}), y_{real}))] \quad (18)$$

V. Synthetic Sample Generation:
   a. Initialize $S = \emptyset$.
   b. For each class $c \in \{1,2,\ldots,K\}$:
   i. Compute required samples $M_c = n_{target}[c] - n_c$ (19)
   ii. For $i = 1$ to M_c:
   - Sample noise $z \sim p(z)$.
   - Generate synthetic instance $x_{synth} = G(z, c)$.
   - Append (x_{synth}, c) to S.

VI. Balanced Dataset Construction:
   Combine datasets:

$$\mathcal{D}_{bal} = \mathcal{D}'_{train} \cup S \quad (20)$$

   Randomly shuffle $\mathcal{D}_{\text{bal}}$.

VII. Return:
   Balanced training dataset $\mathcal{D}_{\text{bal}}$.

We introduce KFCGAN (Algorithm 2), which integrates Kalman filter–based feature smoothing with a CGAN to generate class-balanced, noise-free training data. The Kalman filter stabilizes feature dynamics, while the CGAN learns class-conditional distributions to synthesize realistic minority samples. The resulting balanced dataset enhances classifier robustness and performance on imbalanced multiclass problems.

4.4 KFCGAN-OVO Methodology

In this framework, we have integrated the OVO data decomposition strategy with our suggested KFCGAN method (Algorithm 2). We have utilized the following pipeline:

(i) Raw noisy data $X \rightarrow$ KF denoising $\rightarrow X'$

(ii) $X' \cup$ CGAN augmentation $\rightarrow$ Augmented $X_{aug} = [X'; G(z,1), G(z,2), \ldots, G(z,C)]$ (21)

(iii) OVO decomposition $\rightarrow C(C\text{-}1)/2$ binary classifiers $\{f_{kl}\}$. For classes $y = \{1,2,\ldots,C\}$, train binary classifier $f_{kl}: \mathbb{R}^d \rightarrow \mathbb{R}$ for every pair $k < l$:

$$f_{kl}(x) = w_{kl}^T x + b_{kl} \quad (22)$$

Only samples where $y_i \in \{k, l\}$, relabeled as:

$y_i = k \rightarrow 1$ (positive class)

$y_i = l \rightarrow 0$ (negative class)

(iv) Final prediction: $h(x) = \arg max_c \sum\{l \neq c\} H(f_{cl}(x) > 0)$ (23)

Figure 2 displays the suggested methodology. We began by doing some basic preprocessing of multiclass imbalanced data to resolve missing data, convert categorical data into numbers, and use standard scaling to make the data normal. Then, the dataset is split into two parts: a test set and a training set. The training set is delivered to the CGAN for data augmentation after being smoothed using the Kalman filter. The rebalanced training set is converted into binary data sets using the OVO approach, and these are used to train each binary classifier, such as Classification and Regression Tree (CART) (Dikananda et al., 2022; Lin & Fan, 2019) or Support Vector Machine (SVM) (Batuwita & Palade, 2013; Yu et al., 2025). The results from each classifier are then combined. The binary voting strategy is regarded as the aggregation approach in this study since it is the most straightforward yet effective aggregating method. Each class receives a certain number of votes, and the class with the largest number of votes is the predicted class. Here, we employed stratified 10-fold cross-validation, a technique that divides data sets into ten folds while preserving the original class distribution in each fold to produce more effective and accurate performance estimates for imbalanced datasets. An OVO classifier is used to perform data decomposition in each fold and trains the base classifiers, CART/SVM, accordingly. The results are analyzed using the performance metrics defined in Section 5.

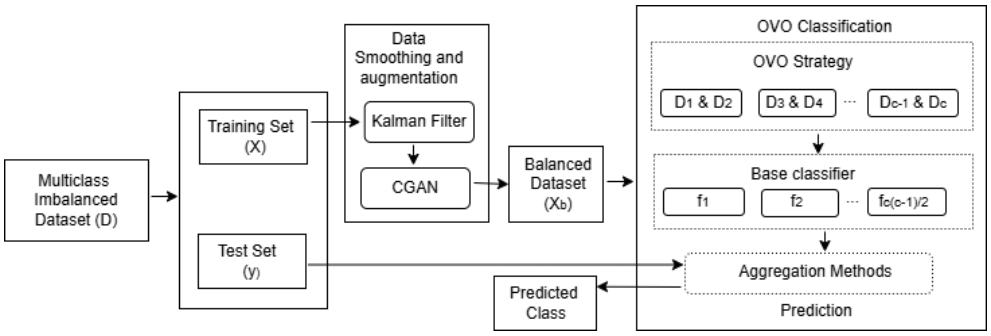


Figure 2. Methodology.

5. Performance Metrics

We looked at how well multiclass classification works when there are imbalances in data sets using Cohen's Kappa (κ), Mean Average Value Accuracy (MAVA), Geometric Mean (MG), Mean F-Measure (MFM), and Macro-Precision (Pmacro). The studies are conducted alongside Pmacro, MAVA, MG, MFM, and Kappa, providing a thorough and imbalance-aware assessment of multiclass classifiers. Let C denote the total number of classes in the dataset.

Pmacro: The average precision across all classes is measured by macro-precision (Pmacro), which is defined as

$$Pmacro = \frac{1}{C}\sum_{i=1}^{C} P_i, \text{ where } P_i = \frac{TP_i}{TP_i+FP_i} \quad (24)$$

Here, *TP* is True Positive, and *FP* is False Positive (Sokolova & Lapalme, 2009).

MAVA: It calculates the average accuracy per class to guarantee that every class is given the same weight (Fernández et al., 2018):

$$MAVA = \frac{1}{C}\sum_{i=1}^{C} \frac{TP_i}{TP_i+FN_i} \quad (25)$$

MG: It calculates the geometric mean of class-wise recalls to assess balanced recognition of all classes (Kubat, 1997):

$$MG = \left(\prod_{i=1}^{C} Recall_i\right)^{\frac{1}{C}}, \text{ where } Recall_i = \frac{TP_i}{TP_i+FN_i} \quad (26)$$

MFM: It gives the average of class-wise F1-scores and is written as follows (Sokolova & Lapalme, 2009):

$$MFM = \frac{1}{C}\sum_{i=1}^{C} F1_i, \text{ where } F1_i = \frac{2P_i.Recall_i}{P_i+Recall_i} \quad (27)$$

Cohen's Kappa: It measures the degree of agreement between predicted and actual classifications while controlling for chance, which is as follows:

$$k = \frac{p_0-p_e}{1-p_e} \quad (28)$$

where, p_e is the predicted agreement by chance and po is the actual agreement (Cohen, 1960).

6. Results and Discussion

The tests are carried out on the fourteen imbalanced multiclass datasets (see **Table 3**) and then on the augmented datasets using the system configurations and parameter values stated in subsection 4.1. Initially, all the datasets are separated into train (*X*) and test (*y*) sets. The distribution of samples throughout the various classes of all the original datasets is shown in **Figures 3(a)-(n)** (see Appendix A.1), with the red oval indicating noisy samples. The imbalanced datasets are then divided into 10 equal folds using the stratified K-folds cross-validation method; the OVO classifier is trained on 9 of the folds using the CART/SVM classifier as a base classifier, and the classifier is tested on the final one-fold. To address the multiclass data sets' high dimensions and complexity, the OVO approach is employed. **Table 4** indicates that CART's optimal results, on average, for the original imbalanced car dataset were 0.9748 Pmacro, 0.9823 MAVA, 0.9816 MG, and 0.9777 MFM; for the win dataset, 0.9658 Pmacro and 0.9476 MG; and for the nur dataset, 0.9546 MAVA, 0.9546 MFM, and the highest, 0.9959 Kappa. On every performance metric, SVM performed better for the win dataset and attained the highest 0.9813 Pmacro, 0.9763 MAVA,

0.9751 MG, 0.9773 MFM, and 0.9663 Kappa. It obtained the second- and third-best scores on all performance metrics for the dry and nur datasets, respectively.

Table 4. Performance of CART and SVM classifiers on the original imbalanced multiclass datasets (without KFCGAN) across various evaluation metrics.

ID	CART					SVM				
	Pmacro	**MAVA**	**MG**	**MFM**	**Kappa**	**Pmacro**	**MAVA**	**MG**	**MFM**	**Kappa**
win	**0.9658 ± 0.0320**	0.9510 ± 0.0529	**0.9476 ± 0.0565**	0.9534 ± 0.0482	0.9315 ± 0.0747	**0.9813 ± 0.0326**	**0.9763 ± 0.0373**	**0.9751 ± 0.0391**	**0.9773 ± 0.0369**	**0.9663 ± 0.0558**
cmc	0.4657 ± 0.0370	0.4558 ± 0.0327	0.4396 ± 0.0436	0.4572 ± 0.0347	0.1920 ± 0.0421	0.5287 ± 0.0547	0.5173 ± 0.0508	0.4963 ± 0.0601	0.5185 ± 0.0524	0.2904 ± 0.0754
bal	0.5794 ± 0.0245	0.5658 ± 0.0366	0.0529 ± 0.1588	0.5712 ± 0.0289	0.6112 ± 0.0590	0.6023 ± 0.0103	0.6528 ± 0.0114	0.0000 ± 0.0000	0.6263 ± 0.0107	0.8190 ± 0.0280
dry	0.9206 ± 0.0053	0.9172 ± 0.0059	0.9161 ± 0.0060	0.9186 ± 0.0055	0.8843 ± 0.0078	**0.9429 ± 0.0051**	**0.9391 ± 0.0057**	**0.9383 ± 0.0058**	**0.9408 ± 0.0054**	**0.9148 ± 0.0077**
gla	0.6887 ± 0.0783	0.7243 ± 0.0906	0.3116 ± 0.3830	0.6816 ± 0.0763	0.5975 ± 0.0986	0.5876 ± 0.1261	0.6168 ± 0.1252	0.0000 ± 0.0000	0.5880 ± 0.1224	0.5693 ± 0.1095
zoo	0.8524 ± 0.1699	0.8829 ± 0.1304	0.5000 ± 0.5000	0.8619 ± 0.1555	0.9222 ± 0.0867	0.8230 ± 0.1541	0.8381 ± 0.1625	0.4000 ± 0.4899	0.8260 ± 0.1599	0.9081 ± 0.0837
car	**0.9748 ± 0.0258**	**0.9823 ± 0.0197**	**0.9816 ± 0.0206**	**0.9777 ± 0.0203**	**0.9810 ± 0.0130**	0.9039 ± 0.0458	0.7662 ± 0.0590	0.7014 ± 0.0876	0.7925 ± 0.0513	0.8217 ± 0.0605
lym	0.7837 ± 0.1319	0.7843 ± 0.1267	0.7105 ± 0.2679	0.7789 ± 0.1310	0.5817 ± 0.2325	0.7381 ± 0.1798	0.7357 ± 0.1593	0.4396 ± 0.4419	0.7302 ± 0.1684	0.7210 ± 0.1323
wqr	0.3628 ± 0.0427	0.3659 ± 0.0300	0.0000 ± 0.0000	0.3560 ± 0.0266	0.4201 ± 0.0451	0.3104 ± 0.0290	0.2882 ± 0.0232	0.0000 ± 0.0000	0.2889 ± 0.0259	0.3856 ± 0.0657
yes	0.5545 ± 0.0682	0.5229 ± 0.0760	0.2126 ± 0.2633	0.5193 ± 0.0682	0.4347 ± 0.0314	0.5797 ± 0.0770	0.5372 ± 0.0800	0.0000 ± 0.0000	0.5406 ± 0.0719	0.4761 ± 0.0664
abs	0.2351 ± 0.0247	0.2356 ± 0.0265	0.0000 ± 0.0000	0.2309 ± 0.0242	0.3269 ± 0.0416	0.2462 ± 0.0372	0.2405 ± 0.0288	0.0000 ± 0.0000	0.2322 ± 0.0288	0.2915 ± 0.0481
wqw	0.4289 ± 0.0515	0.4175 ± 0.0485	0.0494 ± 0.1482	0.4142 ± 0.0441	0.4480 ± 0.0243	0.3375 ± 0.0726	0.2586 ± 0.0218	0.0000 ± 0.0000	0.2615 ± 0.0240	0.3018 ± 0.0234
aba	0.1592 ± 0.0252	0.1473 ± 0.0249	0.0000 ± 0.0000	0.1468 ± 0.0192	0.1224 ± 0.0144	0.1258 ± 0.0197	0.1310 ± 0.0172	0.0000 ± 0.0000	0.1228 ± 0.0163	0.1567 ± 0.0250
nur	0.9548 ± 0.0809	**0.9546 ± 0.0794**	0.7952 ± 0.3977	**0.9546 ± 0.0800**	**0.9959 ± 0.0022**	**0.9315 ± 0.0812**	**0.8335 ± 0.0709**	0.6720 ± 0.3380	**0.8660 ± 0.0740**	**0.9468 ± 0.0047**

In contrast, the Kalman filter smoothed the training set X for every dataset to eliminate noisy samples from it. **Table 5** displays the percentage of noise reduction the Kalman filter achieved for the original datasets. To find the percentage of noise reduction, use the formula as:

$$Noise\ Reduction\ (\%) = \frac{(NR - NK)}{NR} \times 100 \tag{29}$$

where, NR is the noise that was measured in the raw dataset, and NK is the noise that was recorded after the Kalman filter was applied to smooth the dataset. The outcome indicates that the Kalman filter refined the datasets, removing a minimum of 85% of noisy samples. The CGAN is used to make samples of the minority classes once it has been trained on X. This step is done to make sure that each imbalanced dataset has an equal number of samples from each class. The discriminator checks that the samples from the minority class are original once the generator makes them.

Table 5. The noise reduction (in percentage) was performed by the Kalman filter on the original datasets.

Datasets	**win**	**cmc**	**bal**	**dry**	**gla**	**zoo**	**car**	**lym**	**wqr**	**yes**	**abs**	**wqw**	**aba**	**nur**
% Noise Reduction	91.0	91.2	84.8	92.8	89.3	88.3	87.0	90.5	92.0	91.9	91.0	92.1	88.4	85.6

Table 6. Performance of CART and SVM classifiers on the imbalanced multiclass datasets using KFCGAN-OVO approach across various evaluation metrics.

ID	CART					SVM				
	Pmacro	**MAVA**	**MG**	**MFM**	**Kappa**	**Pmacro**	**MAVA**	**MG**	**MFM**	**Kappa**
win	0.9875 ± 0.0191	0.9863 ± 0.0210	0.9856 ± 0.0220	0.9860 ± 0.0214	0.9789 ± 0.0322	0.9833 ± 0.0204	0.9815 ± 0.0227	0.9806 ± 0.0238	0.9812 ± 0.0230	0.9721 ± 0.0342
cmc	0.9801 ± 0.0081	0.9799 ± 0.0082	0.9797 ± 0.0082	0.9798 ± 0.0082	0.9698 ± 0.0122	**0.9922 ± 0.0053**	**0.9920 ± 0.0054**	**0.9920 ± 0.0055**	**0.9920 ± 0.0054**	**0.9881 ± 0.0081**
bal	0.8154 ± 0.0419	0.8113 ± 0.0415	0.8083 ± 0.0415	0.8111 ± 0.0414	0.7170 ± 0.0618	0.7853 ± 0.0328	0.7612 ± 0.0353	0.7549 ± 0.0368	0.7658 ± 0.0342	0.6421 ± 0.0529
dry	**0.9983 ± 0.0006**	**0.9983 ± 0.0006**	**0.9983 ± 0.0006**	**0.9983 ± 0.0006**	**0.9980 ± 0.0008**	**0.9972 ± 0.0013**	**0.9972 ± 0.0013**	**0.9972 ± 0.0013**	**0.9972 ± 0.0013**	**0.9967 ± 0.0015**
gla	0.9694 ± 0.0244	0.9643 ± 0.0292	0.9607 ± 0.0336	0.9607 ± 0.0336	0.9578 ± 0.0336	0.9766 ± 0.0256	0.9744 ± 0.0280	0.9721 ± 0.0307	0.9732 ± 0.0296	0.9685 ± 0.0347
zoo	0.8166 ± 0.0662	0.7821 ± 0.0740	0.7490 ± 0.0872	0.7784 ± 0.0743	0.7481 ± 0.0857	0.9167 ± 0.0107	0.7907 ± 0.0617	0.7515 ± 0.0760	0.8036 ± 0.0570	0.7557 ± 0.0715
car	0.9828 ± 0.0063	0.9826 ± 0.0063	0.9825 ± 0.0064	0.9826 ± 0.0064	0.9769 ± 0.0085	0.9252 ± 0.0042	0.8930 ± 0.0086	0.8832 ± 0.0104	0.8944 ± 0.0086	0.8573 ± 0.0114
lym	0.8096 ± 0.0524	0.8049 ± 0.0470	0.7728 ± 0.0545	0.7996 ± 0.0466	0.7400 ± 0.0633	0.8198 ± 0.0476	0.7937 ± 0.0402	0.7337 ± 0.0743	0.7785 ± 0.0473	0.7242 ± 0.0530
wqr	0.7876 ± 0.0153	0.7834 ± 0.0174	0.7534 ± 0.0203	0.7846 ± 0.0162	0.7401 ± 0.0209	0.8985 ± 0.0065	0.8615 ± 0.0106	0.8318 ± 0.0168	0.8649 ± 0.0117	0.8536 ± 0.0111
yes	0.8442 ± 0.0091	0.8329 ± 0.0109	0.8149 ± 0.0158	0.8360 ± 0.0106	0.8143 ± 0.0123	0.8536 ± 0.0075	0.8231 ± 0.0106	0.8067 ± 0.0129	0.8335 ± 0.0094	0.8035 ± 0.0116
abs	0.8757 ± 0.0101	0.8653 ± 0.0111	0.8363 ± 0.0173	0.8681 ± 0.0109	0.8579 ± 0.0115	0.8979 ± 0.0114	0.8605 ± 0.0111	0.8312 ± 0.0184	0.8647 ± 0.0120	0.8528 ± 0.0115
wqw	0.8121 ± 0.0094	0.8064 ± 0.0103	0.7813 ± 0.0135	0.8079 ± 0.0101	0.7741 ± 0.0120	0.8797 ± 0.0082	0.8245 ± 0.0101	0.7972 ± 0.0128	0.8318 ± 0.0094	0.7953 ± 0.0118
aba	0.8346 ± 0.0062	0.8240 ± 0.0056	0.7763 ± 0.0085	0.8280 ± 0.0055	0.8174 ± 0.0059	0.8550 ± 0.0065	0.8302 ± 0.0053	0.7872 ± 0.0081	0.8386 ± 0.0054	0.8240 ± 0.0055
nur	**0.9915 ± 0.0019**	**0.9915 ± 0.0019**	**0.9914 ± 0.0019**	**0.9915 ± 0.0019**	**0.9894 ± 0.0024**	0.9288 ± 0.0061	0.9261 ± 0.0064	0.9236 ± 0.0069	0.9270 ± 0.0063	0.9076 ± 0.0080

To achieve the optimum results, the discriminator and generator losses are kept to a maximum and minimum, respectively. Subsection 3.1 covers the overall logic behind the CGAN. To accomplish our goals, refer to Algorithm 2; the proposed method (KFCGAN) integrates CGAN with the Kalman filter. **Figures 4(a)-(n)** (see Appendix A.1) illustrate the samples' distribution of the classes for the balanced (resampled) datasets generated by the KFCGAN method. The balanced datasets are then divided into 10 equal folds using the stratified K-folds cross-validation method; the OVO classifier is trained on 9 of the folds using the CART/SVM classifier as a base classifier, and the classifier is tested on the final one-fold. To address the multiclass data sets' high dimensions and complexity, the OVO approach is employed. The results in **Table 6** demonstrate that the KFCGAN-OVO approach substantially enhances the classification performance of both CART and SVM across a diverse set of imbalanced multiclass datasets. CART generally achieves strong and highly stable performance, particularly on datasets such as win, dry, car, abs, aba, and nur, where it consistently records high Pmacro, MAVA, MG, MFM, and Kappa metrics. The highest performance scores attained by the dry and nur datasets are 0.9983 Pmacro, 0.9983 MAVA, 0.9983 MG, 0.9983 MFM, 0.9980 Kappa, and 0.9915 Pmacro, 0.9915 MAVA, 0.9914 MG, 0.9915 MFM, and 0.9894 Kappa, respectively. SVM, on the other hand, works better on a number of datasets, such as cmc, zoo, wqr, and yes, which suggests that it works better when the decision boundaries are more complicated or not linear. The cmc and nur datasets got the best performance scores: 0.9922 Pmacro, 0.9920 MAVA, 0.9920 MG, 0.9920 MFM, and 0.9881 Kappa for cmc and 0.9972 Pmacro, 0.9972 MAVA, 0.9972 MG, 0.9972 MFM, and 0.9967 Kappa for nur. In general, the results show that SVM gets a lot of help from KFCGAN-OVO on some datasets, although CART does well across most scenarios. This means that the KFCGAN augmentation method works well to fix class imbalance and helps both classifiers get results that

are competitive and dependable. However, CART is still a little better overall at being consistent and strong when working with the augmented multiclass imbalanced datasets. The KFCGAN-OVO method that was suggested does not use Euclidean distance or the selection of k-nearest neighbors. It meets the first research objective (RO1).

6.1 Ablation Study

Performance evaluation: **Table 7** shows a comparison of the performance of KFCGAN-OVO, OCH-SMOTE, SOMM, and SMOTE-CLS with CART as the basis classifier. The findings show that, while some trends do appear, no one resampling approach is always better than the others across all the multiclass imbalanced datasets that were tested. When you use KFCGAN-OVO with CART as the base classifier, you always get high-quality classification results. This shows how well it can handle problems with class imbalance. It performs better when accuracy is more important than speed, but it takes a lot longer to compute (in seconds). SMOLTE-CLS is usually one of the better approaches since it gives strong and stable results, even if it takes a lot of computing power. SOMM is the best lightweight method for time-sensitive applications since it is the most dependable and effective, has competitive accuracy, and takes a short amount of time to compute. OCH-SMOTE occasionally doesn't work well with all datasets and doesn't always act the same way, even if it takes less time to compute. Overall, SMOTE-CLS strikes a good compromise between speed and accuracy, SOMM is a fast and reliable option, while KFCGAN-OVO is the best choice if you need better classification performance and don't mind spending more on computing power. **Table 8** also demonstrates how well KFCGAN-OVO, OCH-SMOTE, SOMM, and SMOTE-CLS do when SVM is used as a basis classifier. KFCGAN-OVO consistently outperformed or matched the latest resampling techniques across most datasets. The results indicate that it is more stable, robust, and effective at handling complex multiclass imbalance issues. The method achieves higher Pmacro, MAVA, MG, MFM, and Kappa scores in almost all cases, particularly on challenging datasets such as cmc, dry, gla, wqr, yes, abs, wqw, aba, and nur, where competing techniques like OCH-SMOTE and SOMM exhibit substantial performance degradation. Although SOMM and SMOTE-CLS occasionally deliver competitive or slightly higher scores on a few datasets, their performance is far less consistent, and they frequently fail on highly skewed distributions where KFCGAN-OVO maintains strong predictive capability. While the computational cost of KFCGAN- OVO is higher due to the CGAN and Kalman filtering pipeline, the significant improvement in classification results justifies this trade-off. Overall, the results demonstrate that KFCGAN-OVO provides a highly reliable and more effective solution for multiclass imbalanced learning, beating the existing methods across data domains and imbalance severities. The comparative performance evaluation study demonstrates its reliability and effectiveness on several multiclass imbalanced datasets. These exercises answer the second research objective (RO2).

Table 7. Comparison of the performance of KFCGAN-OVO with the latest methods using CART as the base classifier.

ID	Resampling techniques	CART					Computing time (Sec.)
		Pmacro	MAVA	MG	MFM	Kappa	
win	KFCGAN-OVO	**0.9875 ± 0.0191**	**0.9863 ± 0.0210**	**0.9856 ± 0.0220**	**0.9860 ± 0.0214**	**0.9789 ± 0.0322**	**1.88 ± 0.00**
	OCH-SMOTE	0.9616 ± 0.0299	0.9444 ± 0.0505	0.9459 ± 0.0496	0.9475 ± 0.0457	0.9230 ± 0.0712	0.02 ± 0.00
	SOMM	0.9491 ± 0.0327	0.9387 ± 0.0431	0.9335 ± 0.0494	0.9376 ± 0.0443	0.9084 ± 0.0641	0.02 ± 0.00
	SMOTE-CLS	0.9539 ± 0.0409	0.9476 ± 0.0449	0.9453 ± 0.0469	0.9479 ± 0.0450	0.9223 ± 0.0675	0.67 ± 0.00
cmc	KFCGAN-OVO	**0.9801 ± 0.0081**	**0.9799 ± 0.0082**	**0.9797 ± 0.0082**	**0.9798 ± 0.0082**	**0.9698 ± 0.0122**	**15.95 ± 0.001**
	OCH-SMOTE	0.4522 ± 0.0198	0.4479 ± 0.0185	0.4552 ± 0.0184	0.4473 ± 0.0192	0.1799 ± 0.0227	0.11 ± 0.01
	SOMM	0.5277 ± 0.0395	0.5263 ± 0.0353	0.5194 ± 0.0402	0.5253 ± 0.0370	0.2894 ± 0.0530	0.02 ± 0.00
	SMOTE-CLS	0.5733 ± 0.0480	0.5641 ± 0.0471	0.5616 ± 0.0471	0.5651 ± 0.0469	0.3460 ± 0.0704	10.99 ± 0.00
bal	KFCGAN-OVO	0.8154 ± 0.0419	0.8113 ± 0.0415	0.8083 ± 0.0415	0.8111 ± 0.0414	0.7170 ± 0.0618	1.02 ± 0.00
	OCH-SMOTE	0.6029 ± 0.0302	0.5865 ± 0.0370	0.4644 ± 0.2571	0.5903 ± 0.0335	0.6156 ± 0.0737	0.07 ± 0.00
	SOMM	0.8224 ± 0.0382	0.8183 ± 0.0395	0.8172 ± 0.0393	0.8188 ± 0.0392	0.7274 ± 0.0594	0.01 ± 0.00
	SMOTE-CLS	0.8297 ± 0.0387	0.8266 ± 0.0389	0.8241 ± 0.0391	0.8260 ± 0.0388	0.7396 ± 0.0583	1.84 ± 0.00
dry	KFCGAN-OVO	**0.9983 ± 0.0006**	**0.9983 ± 0.0006**	**0.9983 ± 0.0006**	**0.9983 ± 0.0006**	**0.9980 ± 0.0008**	**154.90 ± 0.11**
	OCH-SMOTE	0.9177 ± 0.0061	0.9188 ± 0.0066	0.9042 ± 0.0064	0.9180 ± 0.0062	0.8854 ± 0.0077	6.66 ± 0.32
	SOMM	0.9051 ± 0.0048	0.9050 ± 0.0048	0.9033 ± 0.0050	0.9050 ± 0.0048	0.8892 ± 0.0056	1.11 ± 0.10
	SMOTE-CLS	0.9376 ± 0.0056	0.9368 ± 0.0057	0.9358 ± 0.0058	0.9371 ± 0.0056	0.9263 ± 0.0066	56.66 ± 0.19
gla	KFCGAN-OVO	**0.9694 ± 0.0244**	**0.9643 ± 0.0292**	**0.9607 ± 0.0336**	**0.9607 ± 0.0336**	**0.9578 ± 0.0336**	**5.01 ± 0.00**
	OCH-SMOTE	0.6268 ± 0.1215	0.6335 ± 0.1057	0.2355 ± 0.1947	0.6123 ± 0.1137	0.5440 ± 0.0852	0.14 ± 0.01
	SOMM	0.8360 ± 0.0231	0.8313 ± 0.0243	0.8125 ± 0.0308	0.8288 ± 0.0257	0.7973 ± 0.0285	0.01 ± 0.00
	SMOTE-CLS	0.8691 ± 0.0599	0.8622 ± 0.0503	0.8352 ± 0.0766	0.8561 ± 0.0573	0.8336 ± 0.0605	1.88 ± 0.07
zoo	KFCGAN-OVO	0.8166 ± 0.0662	0.7821 ± 0.0740	0.7490 ± 0.0872	0.7784 ± 0.0743	0.7481 ± 0.0857	5.20 ± 0.01
	OCH-SMOTE	0.8155 ± 0.1721	0.8376 ± 0.1444	0.5563 ± 0.4270	0.8222 ± 0.1612	0.9090 ± 0.0838	0.07 ± 0.01
	SOMM	0.9895 ± 0.0182	0.9864 ± 0.0220	0.9849 ± 0.0243	0.9865 ± 0.0225	0.9839 ± 0.0267	0.01 ± 0.00
	SMOTE-CLS	0.9971 ± 0.0086	0.9971 ± 0.0086	0.9969 ± 0.0094	0.9968 ± 0.0095	0.9960 ± 0.0121	1.89 ± 0.06
car	KFCGAN-OVO	0.9828 ± 0.0063	0.9826 ± 0.0063	0.9825 ± 0.0064	0.9826 ± 0.0064	0.9769 ± 0.0085	10.15 ± 0.05
	OCH-SMOTE	0.9853 ± 0.0137	0.9523 ± 0.0491	0.9862 ± 0.0108	0.9653 ± 0.0353	0.9722 ± 0.0202	0.20 ± 0.01
	SOMM	0.9813 ± 0.0035	0.9812 ± 0.0035	0.9810 ± 0.0036	0.9812 ± 0.0035	0.9749 ± 0.0047	0.03 ± 0.00
	SMOTE-CLS	0.9943 ± 0.0038	0.9943 ± 0.0039	0.9942 ± 0.0039	0.9943 ± 0.0039	0.9923 ± 0.0052	5.50 ± 0.00
lym	KFCGAN-OVO	0.8096 ± 0.0524	0.8049 ± 0.0470	0.7728 ± 0.0545	0.7996 ± 0.0466	0.7400 ± 0.0633	1.87 ± 0.03
	OCH-SMOTE	0.7831 ± 0.1585	0.7843 ± 0.1474	0.7161 ± 0.2083	0.7775 ± 0.1567	0.5869 ± 0.2769	0.04 ± 0.01
	SOMM	0.9108 ± 0.0310	0.9076 ± 0.0300	0.9006 ± 0.0343	0.9063 ± 0.0315	0.8766 ± 0.0407	0.01 ± 0.00
	SMOTE-CLS	0.9393 ± 0.0298	0.9319 ± 0.0308	0.9260 ± 0.0355	0.9312 ± 0.0314	0.9096 ± 0.0401	0.95 ± 0.00
wqr	KFCGAN-OVO	0.7876 ± 0.0153	0.7834 ± 0.0174	0.7534 ± 0.0203	0.7846 ± 0.0162	0.7401 ± 0.0209	17.74 ± 0.04
	OCH-SMOTE	0.3711 ± 0.0483	0.3728 ± 0.0572	0.4137 ± 0.0824	0.3643 ± 0.0414	0.4140 ± 0.0419	0.59 ± 0.14
	SOMM	0.7673 ± 0.0110	0.7651 ± 0.0133	0.7551 ± 0.0131	0.7655 ± 0.0123	0.7181 ± 0.0159	0.08 ± 0.00
	SMOTE-CLS	0.8117 ± 0.0164	0.8053 ± 0.0196	0.7920 ± 0.0209	0.8066 ± 0.0181	0.7663 ± 0.0237	6.24 ± 0.03
yes	KFCGAN-OVO	0.8442 ± 0.0091	0.8329 ± 0.0109	0.8149 ± 0.0158	0.8360 ± 0.0106	0.8143 ± 0.0123	18.19 ± 0.01
	OCH-SMOTE	0.5315 ± 0.0741	0.5314 ± 0.0716	0.3832 ± 0.1197	0.5123 ± 0.0607	0.4458 ± 0.0540	1.29 ± 0.21
	SOMM	0.7864 ± 0.0176	0.7836 ± 0.0199	0.7631 ± 0.0228	0.7835 ± 0.0187	0.7595 ± 0.0224	0.07 ± 0.00
	SMOTE-CLS	0.9196 ± 0.0091	0.9207 ± 0.0082	0.9079 ± 0.0100	0.9190 ± 0.0084	0.9119 ± 0.0092	5.32 ± 0.01
abs	KFCGAN-OVO	0.8757 ± 0.0101	0.8653 ± 0.0111	0.8363 ± 0.0173	0.8681 ± 0.0109	0.8579 ± 0.0115	14.78 ± 0.13
	OCH-SMOTE	0.2374 ± 0.0402	0.2368 ± 0.0356	0.0702 ± 0.0347	0.2329 ± 0.0357	0.3152 ± 0.0549	2.46 ± 0.43
	SOMM	0.8207 ± 0.0096	0.8170 ± 0.0104	0.7721 ± 0.0176	0.8168 ± 0.0098	0.8069 ± 0.0104	0.12 ± 0.00
	SMOTE-CLS	0.9189 ± 0.0101	0.9166 ± 0.0103	0.9059 ± 0.0136	0.9144 ± 0.0110	0.9119 ± 0.0108	3.67 ± 0.01
wqw	KFCGAN-OVO	0.8121 ± 0.0094	0.8064 ± 0.0103	0.7813 ± 0.0135	0.8079 ± 0.0101	0.7741 ± 0.0120	48.31 ± 0.10
	OCH-SMOTE	0.4581 ± 0.0788	0.4634 ± 0.0740	0.5811 ± 0.0289	0.4517 ± 0.0695	0.4497 ± 0.0261	2.22 ± 0.31
	SOMM	0.7714 ± 0.0122	0.7696 ± 0.0113	0.7565 ± 0.0126	0.7702 ± 0.0115	0.7312 ± 0.0132	0.42 ± 0.06
	SMOTE-CLS	0.9364 ± 0.0067	0.9369 ± 0.0059	0.9309 ± 0.0064	0.9352 ± 0.0060	0.9264 ± 0.0069	17.75 ± 0.04
aba	KFCGAN-OVO	0.8346 ± 0.0062	0.8240 ± 0.0056	0.7763 ± 0.0085	0.8280 ± 0.0055	0.8174 ± 0.0059	78.63 ± 0.69
	OCH-SMOTE	0.1418 ± 0.0193	0.1440 ± 0.0133	0.0512 ± 0.0297	0.1377 ± 0.0123	0.1267 ± 0.0109	27.17±1.67
	SOMM	0.9830 ± 0.0044	0.9829 ± 0.0044	0.9827 ± 0.0045	0.9828 ± 0.00	0.9771 ± 0.0059	0.03 ± 0.00
	SMOTE-CLS	0.9175 ± 0.0059	0.9221 ± 0.0062	0.8991 ± 0.0087	0.9186 ± 0.0060	0.9192 ± 0.0065	19.82 ± 0.19
nur	KFCGAN-OVO	0.9915 ± 0.0019	0.9915 ± 0.0019	0.9914 ± 0.0019	0.9915 ± 0.0019	0.9894 ± 0.0024	146.51 ± 0.01
	OCH-SMOTE	0.9551 ± 0.0782	0.9540 ± 0.0792	0.9945 ± 0.0052	0.9545 ± 0.0786	0.9959 ± 0.0026	2.46 ± 0.55
	SOMM	0.9937 ± 0.0011	0.9937 ± 0.0012	0.9936 ± 0.0012	0.9937 ± 0.0012	0.9921 ± 0.0014	0.07 ± 0.00
	SMOTE-CLS	0.9980 ± 0.0009	0.9980 ± 0.0009	0.9980 ± 0.0009	0.9980 ± 0.0009	0.9975 ± 0.0011	47.19 ± 0.01

Table 8. Comparison of the performance of KFCGAN-OVO with the latest methods using SVM as the base classifier.

ID	Resampling techniques	SVM					Computing time (Sec.)
		Pmacro	MAVA	MG	MFM	Kappa	
win	KFCGAN-OVO	0.9833 ± 0.0204	0.9815 ± 0.0227	0.9806 ± 0.0238	0.9812 ± 0.0230	0.9721 ± 0.0342	1.88 ± 0.00
	OCH-SMOTE	0.7603 ± 0.1042	0.7419 ± 0.1042	0.7015 ± 0.1123	0.7269 ± 0.0986	0.6041 ± 0.1434	0.03 ± 0.00
	SOMM	0.9875 ± 0.0191	0.9863 ± 0.0210	0.9856 ± 0.0220	0.9860 ± 0.0214	0.9789 ± 0.0322	0.17 ± 0.00
	SMOTE-CLS	0.9880 ± 0.0184	0.9838 ± 0.0252	0.9828 ± 0.0268	0.9848 ± 0.0234	0.9776 ± 0.0342	0.11 ± 0.00
cmc	KFCGAN-OVO	**0.9922 ± 0.0053**	**0.9920 ± 0.0054**	**0.9920 ± 0.0055**	**0.9920 ± 0.0054**	**0.9881 ± 0.0081**	**13.37 ± 0.01**
	OCH-SMOTE	0.5078 ± 0.0478	0.5087 ± 0.0501	0.5114 ± 0.0485	0.5030 ± 0.0484	0.2656 ± 0.0671	0.74 ± 0.01
	SOMM	0.6065 ± 0.0198	0.6042 ± 0.0204	0.5987 ± 0.0187	0.6025 ± 0.0191	0.4062 ± 0.0306	0.81 ± 0.28
	SMOTE-CLS	0.6017 ± 0.0397	0.5955 ± 0.0410	0.5881 ± 0.0452	0.5928 ± 0.0432	0.3931 ± 0.0617	11.66 ± 0.22
bal	KFCGAN-OVO	0.7853 ± 0.0328	0.7612 ± 0.0353	0.7549 ± 0.0368	0.7658 ± 0.0342	0.6421 ± 0.0529	1.36 ± 0.04
	OCH-SMOTE	0.7837 ± 0.0326	0.8796 ± 0.0491	0.8615 ± 0.0442	0.7881 ± 0.0475	0.7802 ± 0.0656	0.13 ± 0.00
	SOMM	0.8264 ± 0.0402	0.8032 ± 0.0461	0.8013 ± 0.0461	0.8071 ± 0.0450	0.7048 ± 0.0693	0.23 ± 0.04
	SMOTE-CLS	0.9239 ± 0.0134	0.9005 ± 0.0214	0.8970 ± 0.0226	0.9025 ± 0.0210	0.8506 ± 0.0327	1,97 ± 0.02
dry	KFCGAN-OVO	**0.9972 ± 0.0013**	**0.9972 ± 0.0013**	**0.9972 ± 0.0013**	**0.9972 ± 0.0013**	**0.9967 ± 0.0015**	**156.03 ± 0.32**
	OCH-SMOTE	0.6196 ± 0.0193	0.6231 ± 0.0064	0.5067 ± 0.0179	0.5923 ± 0.0068	0.5539 ± 0.0094	30.60±0.67
	SOMM	0.9543 ± 0.0049	0.9538 ± 0.0049	0.9533 ± 0.0049	0.9539 ± 0.0049	0.9461 ± 0.0057	12.51 ± 0.39
	SMOTE-CLS	0.9497 ± 0.0045	0.9493 ± 0.0047	0.9487 ± 0.0048	0.9494 ± 0.0046	0.9408 ± 0.0055	64.35 ± 0.42
gla	KFCGAN-OVO	**0.9766 ± 0.0256**	**0.9744 ± 0.0280**	**0.9721 ± 0.0307**	**0.9732 ± 0.0296**	**0.9685 ± 0.0347**	**5.03 ± 0.00**
	OCH-SMOTE	0.2893 ± 0.0923	0.3204 ± 0.0977	0.0247 ± 0.0345	0.2666 ± 0.0770	0.1691 ± 0.0928	0.21 ± 0.09
	SOMM	0.8214 ± 0.0438	0.8134 ± 0.0491	0.7859 ± 0.0553	0.8104 ± 0.0462	0.7761 ± 0.0570	0.04 ± 0.00
	SMOTE-CLS	0.8683 ± 0.0447	0.8601 ± 0.0384	0.8404 ± 0.0386	0.8566 ± 0.0369	0.8309 ± 0.0459	1.86 ± 0.05
zoo	KFCGAN-OVO	0.9167 ± 0.0107	0.7907 ± 0.0617	0.7515 ± 0.0760	0.8036 ± 0.0570	0.7557 ± 0.0715	4.91 ± 0.01
	OCH-SMOTE	0.1187 ± 0.0630	0.1501 ± 0.0877	0.0003 ± 0.0008	0.1143 ± 0.0533	0.0480 ± 0.1148	0.10 ± 0.01
	SOMM	1.0000 ± 0.0000	1.0000 ± 0.0000	1.0000 ± 0.0000	1.0000 ± 0.0000	1.0000 ± 0.0000	0.03 ± 0.00
	SMOTE-CLS	1.0000 ± 0.0000	1.0000 ± 0.0000	1.0000 ± 0.0000	1.0000 ± 0.0000	1.0000 ± 0.0000	1.10 ± 0.03
car	KFCGAN-OVO	0.9252 ± 0.0042	0.8930 ± 0.0086	0.8832 ± 0.0104	0.8944 ± 0.0086	0.8573 ± 0.0114	10.49 ± 0.26
	OCH-SMOTE	0.8252 ± 0.0499	0.9341 ± 0.0418	0.9303 ± 0.0293	0.8677 ± 0.0464	0.8582 ± 0.0557	1.20 ± 0.21
	SOMM	0.9534 ± 0.0076	0.9529 ± 0.0075	0.9519 ± 0.0078	0.9525 ± 0.0077	0.9372 ± 0.0100	1.33 ± 0.26
	SMOTE-CLS	0.9788 ± 0.0072	0.9770 ± 0.0084	0.9762 ± 0.0089	0.9770 ± 0.0084	0.9694 ± 0.0111	6.60 ± 0.24
lym	KFCGAN-OVO	0.8198 ± 0.0476	0.7937 ± 0.0402	0.7337 ± 0.0743	0.7785 ± 0.0473	0.7242 ± 0.0530	1.91 ± 0.01
	OCH-SMOTE	0.7855 ± 0.1755	0.7621 ± 0.1638	0.6917 ± 0.2506	0.7622 ± 0.1707	0.5994 ± 0.1867	0.04 ± 0.00
	SOMM	0.9505 ± 0.0308	0.9441 ± 0.0336	0.9392 ± 0.0373	0.9437 ± 0.0341	0.9258 ± 0.0448	0.03 ± 0.00
	SMOTE-CLS	0.9541 ± 0.0286	0.9476 ± 0.0313	0.9428 ± 0.0351	0.9468 ± 0.0320	0.9300 ± 0.0415	0.96 ± 0.00
wqr	KFCGAN-OVO	**0.8985 ± 0.0065**	**0.8615 ± 0.0106**	**0.8318 ± 0.0168**	**0.8649 ± 0.0117**	**0.8536 ± 0.0111**	**16.16 ± 0.38**
	OCH-SMOTE	0.2411 ± 0.0862	0.1850 ± 0.0843	0.0535 ± 0.0541	0.1383 ± 0.0207	0.0524 ± 0.0348	3.24 ± 0.49
	SOMM	0.8081 ± 0.0167	0.8057 ± 0.0174	0.7875 ± 0.0210	0.8051 ± 0.0172	0.7668 ± 0.0206	1.70 ± 0.32
	SMOTE-CLS	0.7703 ± 0.0308	0.7822 ± 0.0250	0.7481 ± 0.0356	0.7719 ± 0.0275	0.7386 ± 0.0302	7.91 ± 0.36
yes	KFCGAN-OVO	**0.8536 ± 0.0075**	**0.8231 ± 0.0106**	**0.8067 ± 0.0129**	**0.8335 ± 0.0094**	**0.8035 ± 0.0116**	**19.43 ± 0.32**
	OCH-SMOTE	0.0768 ± 0.0251	0.1268 ± 0.0537	0.0035 ± 0.0068	0.0653 ± 0.0148	0.0157 ± 0.0238	4.53 ± 0.45
	SOMM	0.8136 ± 0.0152	0.8028 ± 0.0151	0.7859 ± 0.0179	0.8033 ± 0.0145	0.7809 ± 0.0167	1.83 ± 0.37
	SMOTE-CLS	0.7741 ± 0.0182	0.7642 ± 0.0177	0.7433 ± 0.0228	0.7651 ± 0.0179	0.7379 ± 0.0199	7.60 ± 0.27
abs	KFCGAN-OVO	**0.8979 ± 0.0114**	**0.8605 ± 0.0111**	**0.8312 ± 0.0184**	**0.8647 ± 0.0120**	**0.8528 ± 0.0115**	**15.65 ± 0.26**
	OCH-SMOTE	0.0039 ± 0.0118	0.0007 ± 0.0022	0.0000 ± 0.0000	0.0012 ± 0.0037	0.0023 ± 0.0056	4.42 ± 0.49
	SOMM	0.8520 ± 0.0136	0.8547 ± 0.0102	0.8164 ± 0.0177	0.8508 ± 0.0116	0.8467 ± 0.0112	1.63 ± 0.28
	SMOTE-CLS	0.8278 ± 0.0128	0.8276 ± 0.0142	0.7911 ± 0.0207	0.8228 ± 0.0146	0.8178 ± 0.0157	5.45 ± 0.35
wqw	KFCGAN-OVO	**0.8797 ± 0.0082**	**0.8245 ± 0.0101**	**0.7972 ± 0.0128**	**0.8318 ± 0.0094**	**0.7953 ± 0.0118**	**62.83 ± 0.62**
	OCH-SMOTE	0.1921 ± 0.0250	0.1951 ± 0.0474	0.1426 ± 0.0646	0.1656 ± 0.0218	0.0587 ± 0.0182	27.56±0.91
	SOMM	0.8016 ± 0.0150	0.7901 ± 0.0138	0.7677 ± 0.0170	0.7887 ± 0.0144	0.7552 ± 0.0160	20.32 ± 1.10
	SMOTE-CLS	0.7587 ± 0.0102	0.7680 ± 0.0092	0.7331 ± 0.0122	0.7607 ± 0.0095	0.7294 ± 0.0107	47.12 ± 0.59
aba	KFCGAN-OVO	**0.8550 ± 0.0065**	**0.8302 ± 0.0053**	**0.7872 ± 0.0081**	**0.8386 ± 0.0054**	**0.8240 ± 0.0055**	**100.12 ± 0.47**
	OCH-SMOTE	0.0980 ± 0.0157	0.0981 ± 0.0131	0.0117 ± 0.0053	0.0896 ± 0.0084	0.1388 ± 0.0128	48.36±2.21
	SOMM	0.9545 ± 0.0061	0.9535 ± 0.0065	0.9522 ± 0.0068	0.9530 ± 0.0066	0.9380 ± 0.0086	1.30 ± 0.25
	SMOTE-CLS	0.5469 ± 0.0069	0.5605 ± 0.0066	0.4342 ± 0.0142	0.5448 ± 0.0061	0.5442 ± 0.0068	71.94 ± 0.57
nur	KFCGAN-OVO	**0.9288 ± 0.0061**	**0.9261 ± 0.0064**	**0.9236 ± 0.0069**	**0.9270 ± 0.0063**	**0.9076 ± 0.0080**	**191.90 ± 0.78**
	OCH-SMOTE	0.7688 ± 0.0866	0.8355 ± 0.0902	0.9377 ± 0.0095	0.7897 ± 0.0887	0.9166 ± 0.0106	15.31±0.45
	SOMM	0.9715 ± 0.0043	0.9712 ± 0.0044	0.9702 ± 0.0046	0.9708 ± 0.0045	0.9639 ± 0.0055	13.61 ± 0.37
	SMOTE-CLS	0.9799 ± 0.0024	0.9794 ± 0.0026	0.9788 ± 0.0028	0.9792 ± 0.0026	0.9742 ± 0.0032	60.80 ± 0.22

Data complexity analysis: We also investigated the performance of KFCGAN, OCH-SMOTE, SOMM, and SMOTE-CLS in managing the dataset's complexity. The resampled datasets were evaluated for data complexity using the metrics in **Table 9** (see Appendix A.2). These measures are computed using the open-source Problexity Python library (Komorniczak & Ksieniewicz, 2022). **Table 10** lists the data complexity classes and associated ranges of values. Here, 0 indicates a lower value, and 1 is the highest value. The classes are divided into five groups based on the data sets' difficulty: a lower entropy value for the class proportions (C1) signifies a highly imbalanced distribution, resulting in lower data complexity; conversely, a higher entropy value indicates a uniform class distribution and thus higher data complexity; additionally, a larger imbalance ratio (C2) reflects greater imbalance and complexity. A more complex issue is indicated by the higher value of the F1 measure, which lies between 0 and 1 and reflects the overlap of feature values across classes. F1.v, a variant of F1 that employs a "directional vector," displays lower values compared to F1. F2 represents the volume of the overlapping regions. A low value indicates small overlap and small data complexity. Higher F3 and F4 scores suggest higher difficulties. The larger values of the dimensionality measures T2, T3, and T4 indicate higher data complexity. Higher values indicate increased complexity for all linearity-focused complexity indicators, including L1, L2, and L3. There are six neighborhood measures, including "Local Set Average Cardinality" (LSC), T1, N1, N2, N3, and N4. Larger values for T1, N1, N2, N3, and N4 indicate higher complexity. Conversely, a smaller LSC value signifies greater complexity. Density, cluster coefficient (ClsCoef), and Hubs measures are used to quantify the complexity of a network. High values of density, ClsCoef, and Hubs scores indicate high complexity (Eberlein et al., 2025).

Table 10. Range of values for various classes of data complexity.

S. No.	Complexity class	Complexity range
1.	Extremely Difficult	0.71 to 1.00
2.	Difficult	0.51 to 0.70
3.	Medium	0.31 to 0.50
4.	Simple	0.11 to 0.30
5.	Extremely Simple	0.00 to 0.10

For **Tables 11-15**, refer to the Appendix A.2. **Table 11** displays the computed values of all data complexity measures for each of the original multiclass datasets given in **Table 3**. The last row of the table contains the overall complexity score for each dataset. Thus, for all multiclass datasets resampled (or augmented) using the KFCGAN, OCH-SMOTE, SOMM, and SMOTE-CLS methods, **Tables 12-15** provide the values of all data complexity measures, respectively. Again, each table's last row contains the dataset's overall complexity score. The overall data complexity scores for each dataset are shown in **Table 16**, which enables a comparison of how well KFCGAN, OCH-SMOTE, SOMM, and SMOTE-CLS handle data complexity. The findings show that the KFCGAN is more effective in reducing the overall data complexity of the datasets and outperforms the others in handling the complexity of massive, intricate, and highly imbalanced multiclass datasets. The results of our proposed KFCGAN method provide an answer to the third research objective (RO3).

Table 16. Comparison of KFCGAN, OCH-SMOTE, SOMM, and SMOTE-CLS in mitigating the data complexity of multiclass datasets.

Resampling techniques	Overall data complexity scores													
	win	**cmc**	**bal**	**dry**	**gla**	**zoo**	**car**	**lym**	**wqr**	**yes**	**abs**	**wqw**	**aba**	**nur**
NO Sampling	22.2	47.9	45.8	15.7	24.1	24.0	39.0	31.8	37.4	30.2	19.4	39.3	29.9	32.7
KFCGAN	**13.1**	21.0	**28.1**	**9.5**	**14.3**	18.8	**17.4**	21.5	**19.4**	**17.1**	**14.3**	**17.7**	**15.8**	**21.0**
OCH-SMOTE	17.9	41.9	38.9	18.9	21.7	15.2	32.4	**18.9**	30.8	23.9	19.1	31.3	21.4	24.7
SOMM	20.1	47.0	43.5	21.0	22.8	20.0	33.4	20.0	28.9	20.0	20.2	28.1	20.2	25.6
SMOTE-CLS	23.4	**20.4**	41.9	15.0	20.4	**14.4**	33.3	19.6	29.1	24.0	20.3	31.3	20.0	25.7

Computational complexity analysis: The results in **Table 7** demonstrate that KFCGAN-OVO takes the longest to execute, from 1.02 seconds on smaller datasets like bal to 154.90 seconds on larger datasets like dry and 146.51 seconds on nur. This extra work is a lot more than what classic resampling methods like OCH-SMOTE or SOMM need. For many datasets, they can finish in less than a second. The CGAN is utilized for data augmentation and the OVO classifier is used for multiclass discrimination at the same time, which makes the run time longer. KFCGAN-OVO works well and consistently on a wide range of datasets, making it a good candidate for cases where accuracy is more critical than speed, even if it needs more processing resources. **Table 8** also shows that KFCGAN-OVO is far more computationally complex than other methods of resampling. For instance, it takes 156.03 seconds on the dry dataset, which is much longer than OCH-SMOTE (30.60 seconds), SOMM (12.51 seconds), and SMOTE-CLS (64.35 seconds). Likewise, on bigger datasets like nur and aba, the timings are 191.90 s and 100.12 s, respectively. This is because of the extra time needed for GAN-based synthetic data generation and the one-vs-one SVM method. Even though these runtimes are longer, KFCGAN-OVO always performs better on all metrics (Pmacro, MAVA, MG, MFM, Kappa), which means that there is a trade-off between speed and accuracy.

6.2 Wilcoxon Signed-Rank Test

The Wilcoxon signed-rank (Rey & Neuhäuser, 2011) test is used to compare the statistical pairwise performance of the CART and SVM models. The comparative analysis (refer to **Table 17**) reveals that the relative performance of CART and SVM varies across datasets, using the Wilcoxon signed-rank test, effect size measures, and bootstrap 95% confidence intervals. Significant difference ($p < 0.05$) is observed for cmc, car, wqr, abs, wqw, aba, and nur datasets, indicating strong evidence of model superiority in these cases. Among these, the car and nur datasets show large positive effect sizes and strictly positive confidence intervals, demonstrating a clear advantage for SVM. Conversely, datasets such as cmc, wqr, abs, wqw, and aba exhibit large negative effect sizes with entirely confident intervals, confirming the consistent superiority of CART. For the remaining datasets, the p-values and confidence intervals indicate no statistical difference between the two models. Overall, the results highlight that neither CART nor SVM universally dominates; instead, directionally consistent differences are confirmed by both effect size magnitude and bootstrap confidence intervals. **Figure 5(a)-(n)** (see Appendix A.3) shows that the per-fold F1-score distributions across all 14 datasets reveal clear performance differences between CART and SVM. Overall, SVM demonstrates more stable and consistently higher F1-scores on most datasets, indicated by tighter and higher-distribution ranges. CART has more variation and lower F1 scores in some cases, especially when the datasets are more complicated. These results suggest that SVM generally offers superior predictive reliability across diverse data characteristics, while CART may be more sensitive to dataset complexity and variability. The calibration curves (**Figure 6(a)-(n)**, refer to the Appendix A.3) indicate notable differences in the probabilistic reliability of CART and SVM across the data sets. Overall, SVM exhibits smoother and better-aligned calibration with the ideal diagonal reference line for most data sets, reflecting more reliable probability estimates. In contrast, CART frequently displays greater deviations from perfect calibration, suggesting overconfidence or underconfidence in certain probability ranges. These findings highlight SVM's superior ability to produce well-calibrated predictions across diverse data scenarios, while CART's probability estimates tend to be less consistent and more dataset-dependent.

Table 17. Statistical evaluation of CART vs. SVM performance on KFCGAN-OVO–augmented multiclass datasets using Wilcoxon signed-rank tests, effect size metrics, and bootstrap-derived confidence intervals.

Dataset	p-value	Wilcoxon _stat	Paired Cohen's d	Cliff's delta	Mean difference	bootstrap 95% CI
win	0.2500	0.00	-0.621	-0.300	-0.0144	[-0.0287, 0.0000]
cmc	0.0020	0.0	-2.546	-0.980	-0.0474	[-0.0589, -0.0365]
bal	0.4922	20.0	0.199	0.060	0.0076	[-0.0138, 0.0309]
dry	0.5566	21.0	-0.187	-0.220	-0.0002	[-0.0008, 0.0004]
gla	0.4922	20.0	0.366	0.200	0.0161	[-0.0098, 0.0431]

Table 17 continued…

zoo	0.5566	21.0	-0.363	-0.140	-0.0189	[-0.0503, 0.0097]
car	0.0020	0.00	7.228	1.000	0.0954	[0.0870, 0.1027]
lym	0.1934	14.0	-0.543	-0.360	-0.0418	[-0.0859, 0.0039]
wqr	0.0059	2.00	-1.486	-0.760	-0.0297	[-0.0419, -0.0176]
yes	0.0840	10.0	-0.687	-0.240	-0.0079	[-0.0144, -0.0007]
abs	0.0039	1.00	-1.650	-0.540	-0.0177	[-0.0237, -0.0107]
wqw	0.0020	0.00	-3.719	-1.000	-0.0378	[-0.0431, -0.0313]
aba	0.0020	0.00	-4.271	-0.960	-0.0175	[-0.0199, -0.0153]
nur	0.0020	0.00	7.797	1.000	0.0564	[0.0520, 0.0606]

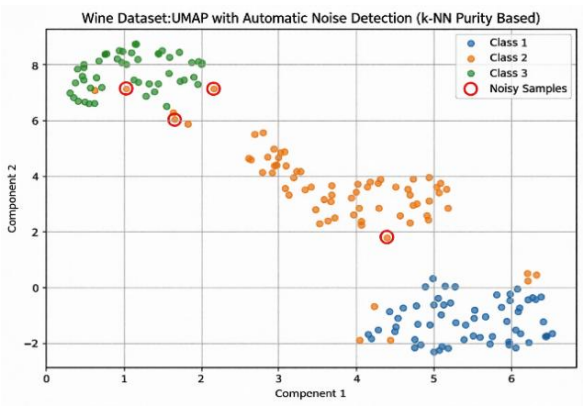
7. Conclusion and Future Directions

In this research, we have proposed a novel KFCGAN-OVO technique to address the issues raised due to imbalanced data distribution, class overlapping, noise, and high dimensionality of data sets. It is a hybrid technique to merge the Kalman filter to filter out the noisy samples, CGAN to rebalance the data sets, and the OVO strategy to tackle the issues of high dimensionality. The findings suggest that the proposed method is more reliable for addressing issues with large and complex multiclass imbalance problems. The results are verified on three fronts: performance, computing time, and data complexity. On these terms, KFCGAN-OVO outperformed the OCH-SMOTE, SOMM, and SMOTE-CLS techniques using CART/SVM as a base classifier. The superiority of KFCGAN-OVO is also tested for the statistical performance of CART vs. SVM on multiclass datasets using the Wilcoxon signed-rank test. The findings highlight SVM's superior ability to produce well-calibrated predictions across diverse data scenarios, while CART's probability estimates tend to be less consistent and more dataset-dependent.

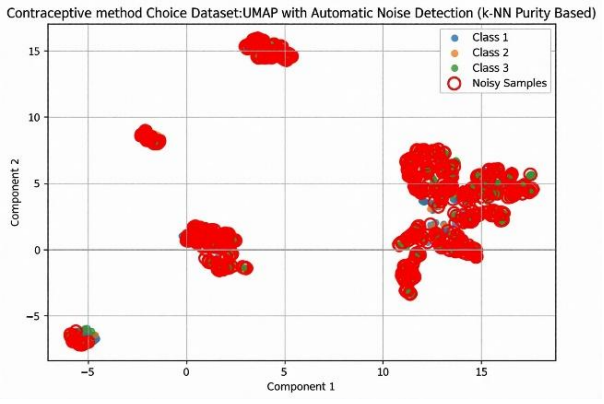
The limitations (see Appendix A.4) of the KFCGAN-OVO approach incurs comparatively high computing costs because it utilizes both CGAN and OVO methods; however, it remains manageable for highly complex and imbalanced multiclass data sets while achieving high performance and better accuracy. In the near future, we will use a different GAN model and filter to reduce computing costs and also enhance the performance of the KFCGAN-OVO method.

Appendix

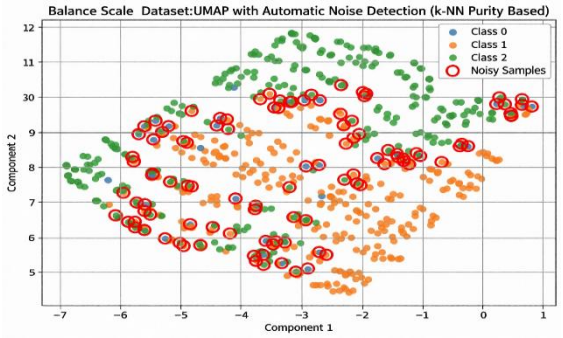
A.1: The samples distribution of the multiclass imbalanced and KFCGAN's augmented data sets.



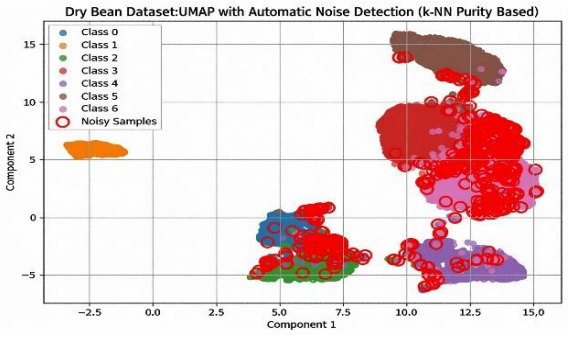
(a)



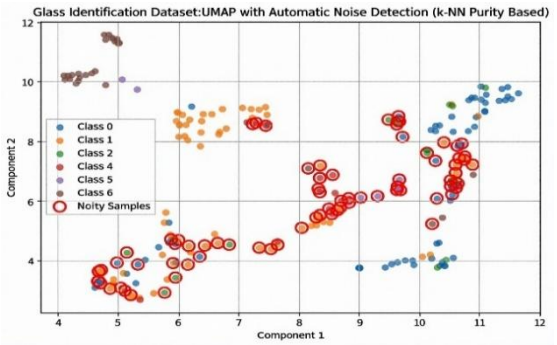
(b)


Balance Scale Dataset:UMAP with Automatic Noise Detection (k-NN Purity Based)
Class 0
Class 1
Class 2
Noisy Samples
Component 1
Component 2

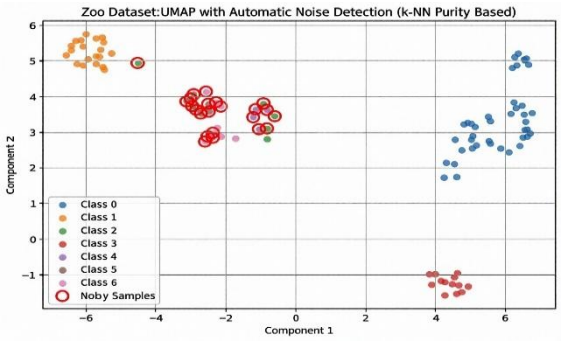
(c)


Dry Bean Dataset:UMAP with Automatic Noise Detection (k-NN Purity Based)
Class 0
Class 1
Class 2
Class 3
Class 4
Class 5
Class 6
Noisy Samples
Component 1
Component 2

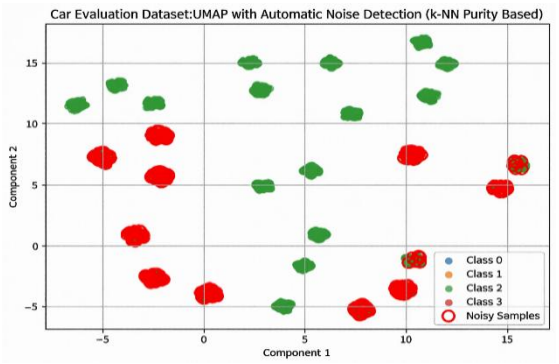
(d)


Glass Identification Dataset:UMAP with Automatic Noise Detection (k-NN Purity Based)
Class 0
Class 1
Class 2
Class 4
Class 5
Class 6
Noisy Samples
Component 1
Component 2

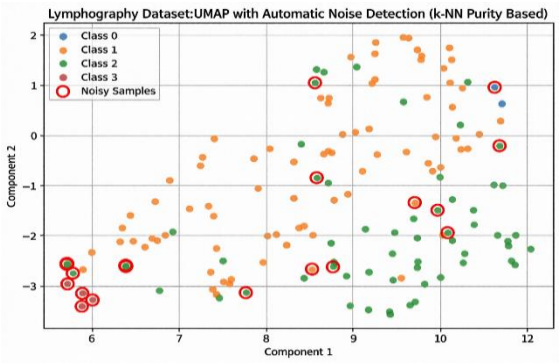
(e)


Zoo Dataset:UMAP with Automatic Noise Detection (k-NN Purity Based)
Class 0
Class 1
Class 2
Class 3
Class 4
Class 5
Class 6
Noby Samples
Component 1
Component 2

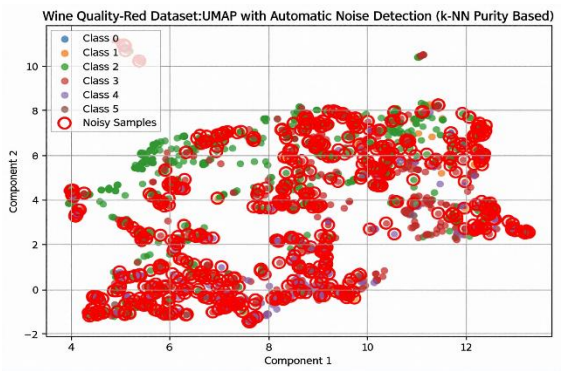
(f)


Car Evaluation Dataset:UMAP with Automatic Noise Detection (k-NN Purity Based)
Class 0
Class 1
Class 2
Class 3
Noisy Samples
Component 1
Component 2

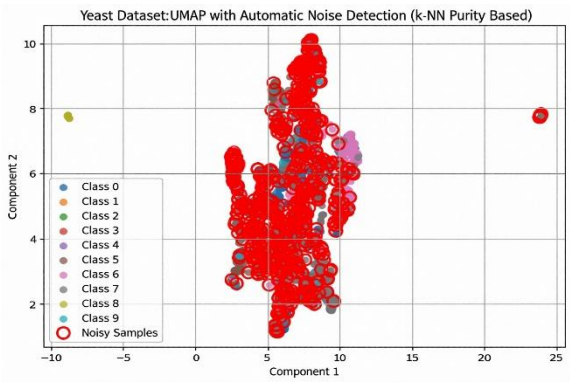
(g)


Lymphography Dataset:UMAP with Automatic Noise Detection (k-NN Purity Based)
Class 0
Class 1
Class 2
Class 3
Noisy Samples
Component 1
Component 2

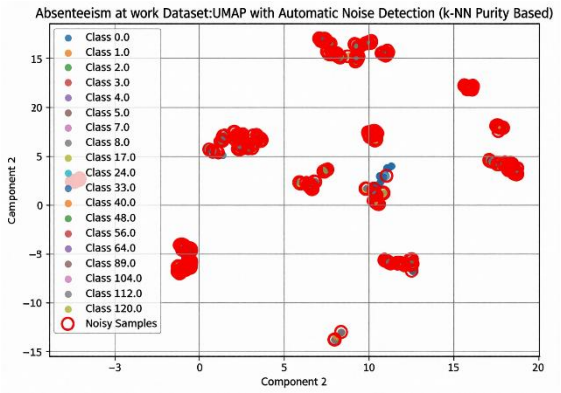
(h)


Wine Quality-Red Dataset:UMAP with Automatic Noise Detection (k-NN Purity Based)
Class 0
Class 1
Class 2
Class 3
Class 4
Class 5
Noisy Samples
Component 1
Component 2

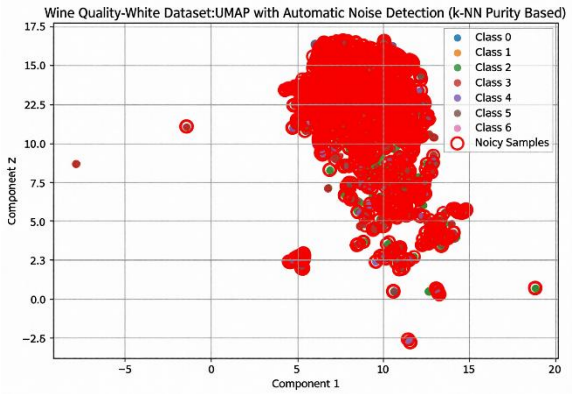
(i)


Yeast Dataset:UMAP with Automatic Noise Detection (k-NN Purity Based)
Class 0
Class 1
Class 2
Class 3
Class 4
Class 5
Class 6
Class 7
Class 8
Class 9
Noisy Samples
Component 1
Component 2

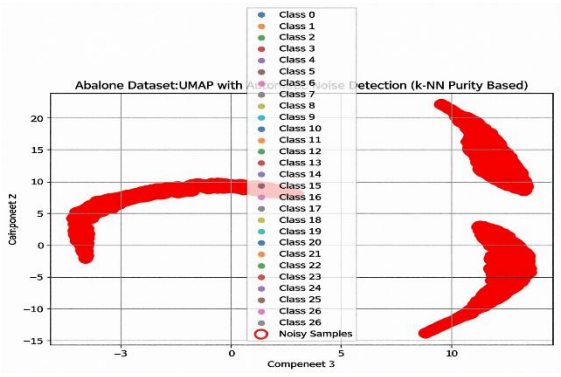
(j)



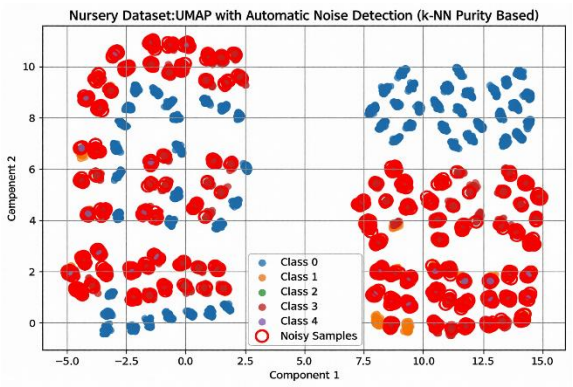
(k)



(l)

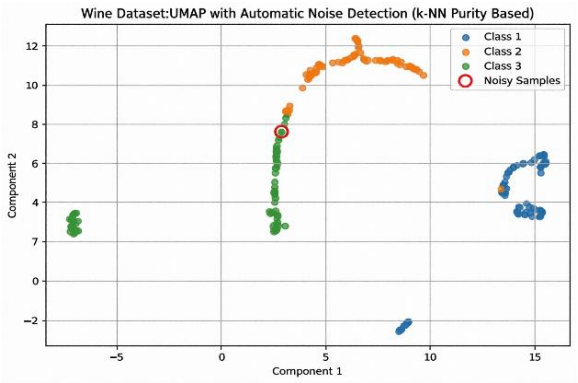


(m)

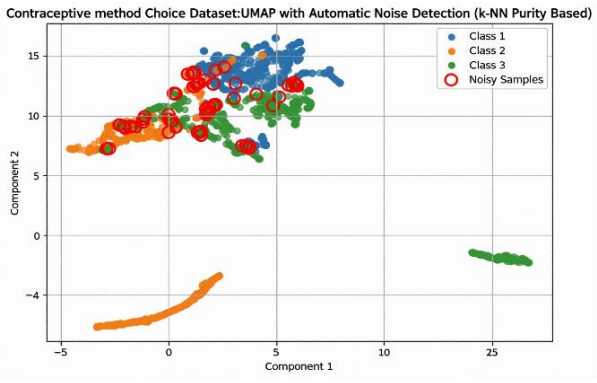


(n)

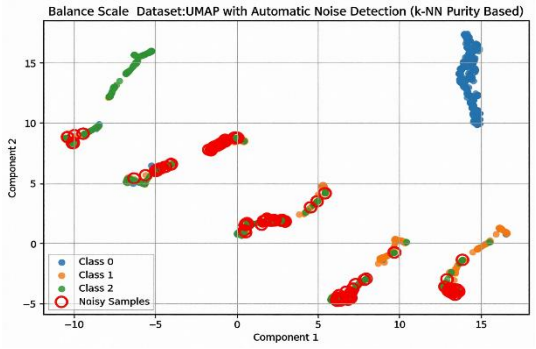
Figure 3. The sample distribution of the classes for the original datasets: (a) win, (b) cmc, (c) bal, (d) dry, (e) gla, (f) zoo, (g) car, (h) lym, (i) wqr, (j) yes, (k) abs, (l) wqw, (m) aba, and (n) nur. In figures, the red oval indicates noisy samples.



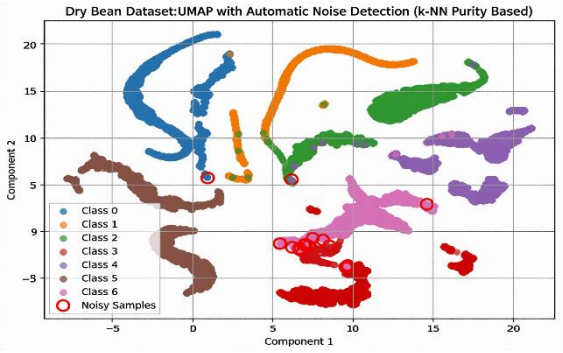
(a)



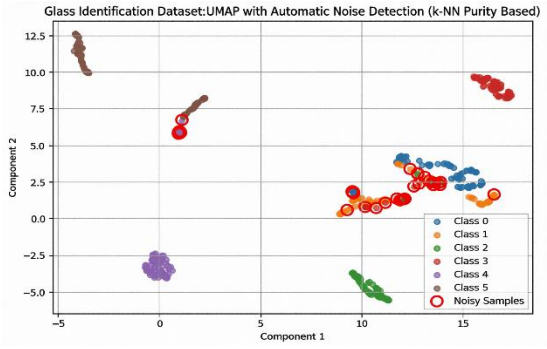
(b)


Balance Scale Dataset:UMAP with Automatic Noise Detection (k-NN Purity Based)
Component 1
Component 2
Class 0
Class 1
Class 2
Noisy Samples

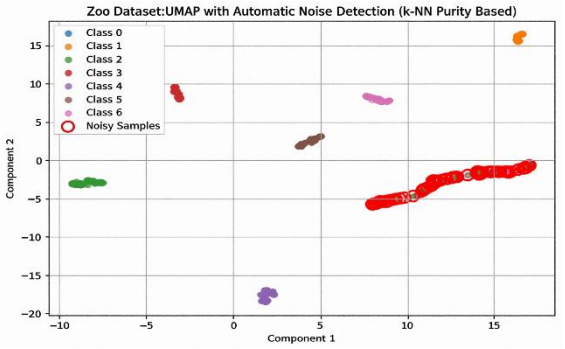
(c)


Dry Bean Dataset:UMAP with Automatic Noise Detection (k-NN Purity Based)
Component 1
Component 2
Class 0
Class 1
Class 2
Class 3
Class 4
Class 5
Class 6
Noisy Samples

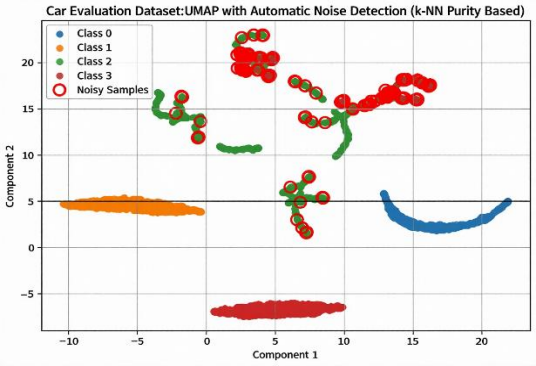
(d)


Glass Identification Dataset:UMAP with Automatic Noise Detection (k-NN Purity Based)
Component 1
Component 2
Class 0
Class 1
Class 2
Class 3
Class 4
Class 5
Noisy Samples

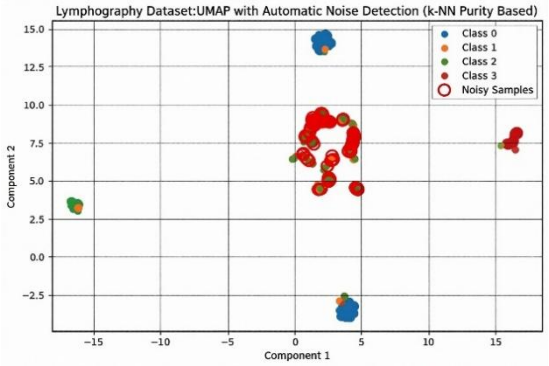
(e)


Zoo Dataset:UMAP with Automatic Noise Detection (k-NN Purity Based)
Component 1
Component 2
Class 0
Class 1
Class 2
Class 3
Class 4
Class 5
Class 6
Noisy Samples

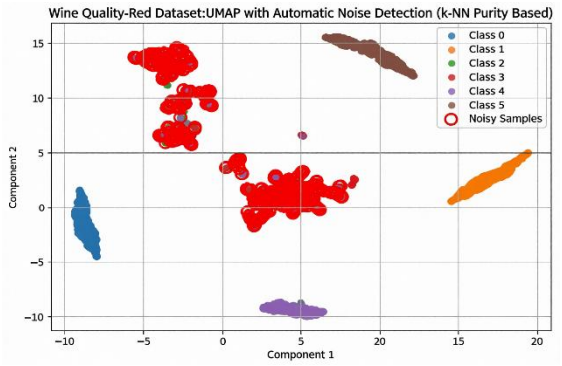
(f)


Car Evaluation Dataset:UMAP with Automatic Noise Detection (k-NN Purity Based)
Component 1
Component 2
Class 0
Class 1
Class 2
Class 3
Noisy Samples

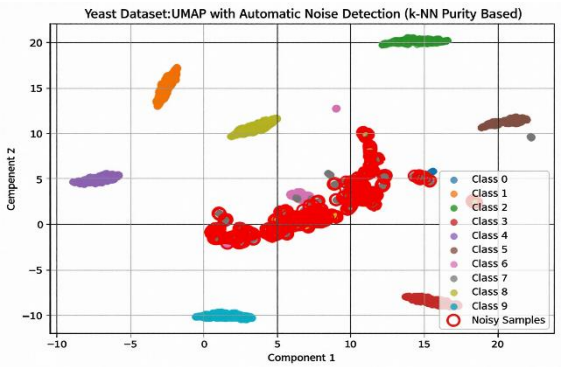
(g)


Lymphography Dataset:UMAP with Automatic Noise Detection (k-NN Purity Based)
Component 1
Component 2
Class 0
Class 1
Class 2
Class 3
Noisy Samples

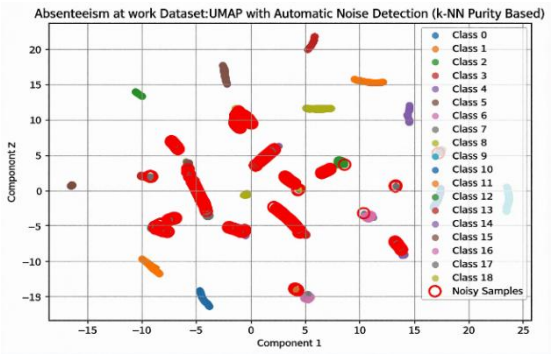
(h)


Wine Quality-Red Dataset:UMAP with Automatic Noise Detection (k-NN Purity Based)
Component 1
Component 2
Class 0
Class 1
Class 2
Class 3
Class 4
Class 5
Noisy Samples

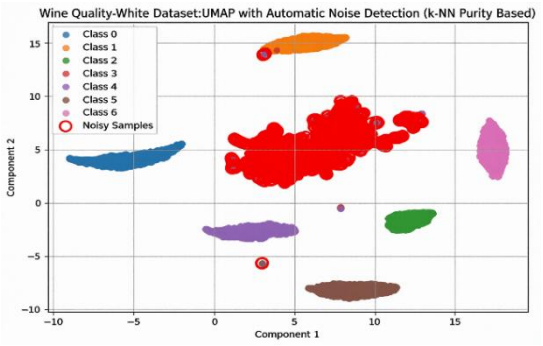
(i)


Yeast Dataset:UMAP with Automatic Noise Detection (k-NN Purity Based)
Component 1
Component 2
Class 0
Class 1
Class 2
Class 3
Class 4
Class 5
Class 6
Class 7
Class 8
Class 9
Noisy Samples

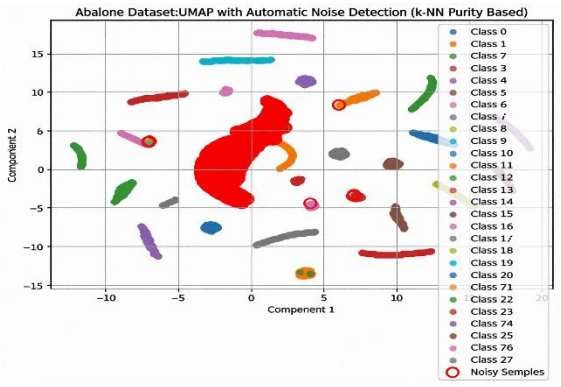
(j)



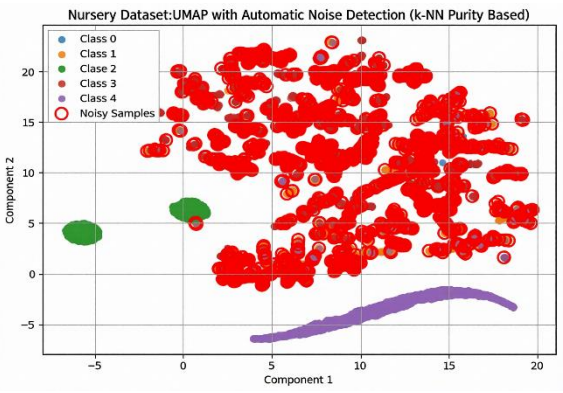
(k)



(l)



(m)



(n)

Figure 4. The sample distribution of the classes for the augmented datasets: (a) win, (b) cmc, (c) bal, (d) dry, (e) gla, (f) zoo, (g) car, (h) lym, (i) wqr, (j) yes, (k) abs, (l) wqw, (m) aba, and (n) nur. In figures, the red oval indicates noisy samples.

A.2: Description of measures and additional results of the data complexity analysis.

Table 9. Description of data complexity measures and their classes (Eberlein et al., 2025).

Class	Description	Data complexity metric	Notation	Value description
Balance	Distribution of samples among the classes	Entropy of Class Proportion	C1	High imbalance is associated with high complexity
		Imbalance Ratio	C2	
Dimensionality	Describe the sparsity of the data	Average Number of Points per Dimension	T2	Large values indicate a more complex problem
		Average Number of Points per PCA	T3	
		Ratio of the PCA Dimension to the Original	T4	
Linearity	Refers the separability of the classes	Sum of the Error Distance by Linear Programming	L1	Large values show high complexity
		Error rate of a linear classifier	L2	
		Non-linearity of a linear classifier	L3	
Neighborhood	Rely on decision boundaries, overlaps, and neighborhood distances.	Fraction of Borderline Points	N1	High values indicate high complexity
		Ratio of Intra/Extra Class Nearest Neighbor Distance	N2	
		Error Rate of the Narest Neighbor	N3	
		Non-linearity of the Narest Neighbor ClassifierN4	N4	
		Fraction of Hypersphere Covering Data	T1	

Table 9 continued…

		Local Set average Cardinality	LSC	High scores indicate simpler problems
Network	The ϵ-NN algorithm is used to convert the data into a graph	Average Density of the Network	Density	High values indicate high complexity
		Clustering Coefficient	ClsCoef	
		Hubs Score	Hubs	
Overlap	Refers to how helpful the information provided is for class separation.	Maximum Fisher's Discriminant Ratio	F1	High values indicate high complexity
		Directional -vector maximum Fisher's discriminant ratio	F1.v	
		Volume of the overlapping region	F2	
		Maximum individual feature efficiency	F3	
		Collective feature efficiency	F4	

Table 11. Data complexity analysis of the original imbalanced multiclass datasets across various measures.

Complexity measures	**Dataset**													
	win	**cmc**	**bal**	**dry**	**gla**	**zoo**	**car**	**lym**	**wqr**	**yes**	**abs**	**wqw**	**aba**	**nur**
Density	0.91	0.94	0.88	0.67	0.79	0.83	0.95	0.92	0.95	0.93	0.76	0.95	0.74	0.97
ClsCoef	0.55	0.43	0.52	0.21	0.38	0.57	0.60	0.76	0.55	0.61	0.18	0.52	0.25	0.66
Hobs	0.79	0.84	0.66	0.53	0.61	0.65	0.72	0.71	0.84	0.81	0.62	0.85	0.56	0.77
L1	0.00	0.18	0.01	0.08	0.02	0.00	0.19	0.00	0.05	0.01	0.01	0.06	0.02	0.02
L2	0.00	0.33	0.07	0.01	0.05	0.00	0.11	0.02	0.07	0.07	0.06	0.08	0.12	0.05
L3	0.00	0.29	0.03	0.01	0.04	0.00	0.10	0.01	0.07	0.06	0.05	0.07	0.11	0.04
T2	0.11	0.01	0.01	0.00	0.17	0.79	0.01	0.69	0.07	0.07	0.01	0.04	0.05	0.00
T3	0.08	0.00	0.01	0.00	0.08	0.30	0.01	0.21	0.05	0.01	0.00	0.02	0.01	0.00
T4	0.74	0.22	1.00	0.06	0.44	0.41	0.97	0.49	0.75	0.11	0.35	0.77	0.25	0.90
F1	0.15	0.85	0.85	0.10	0.38	0.02	0.79	0.32	0.71	0.38	0.18	0.83	0.61	0.57
F1. v	0.03	0.57	0.23	0.03	0.09	0.03	0.43	0.05	0.22	0.14	0.10	0.27	0.25	0.19
F2	0.00	0.77	1.00	0.00	0.00	0.00	0.11	0.04	0.02	0.00	0.04	0.02	0.05	0.10
F3	0.15	0.96	1.00	0.26	0.26	0.05	0.55	0.17	0.68	0.51	0.48	0.70	0.56	0.27
F4	0.00	0.96	1.00	0.16	0.06	0.00	0.21	0.15	0.37	0.29	0.29	0.49	0.38	0.14
N1	0.02	0.19	0.09	0.03	0.05	0.03	0.03	0.04	0.06	0.10	0.04	0.06	0.08	0.02
N2	0.31	0.42	0.37	0.04	0.23	0.23	0.33	0.29	0.27	0.11	0.12	0.26	0.20	0.23
N3	0.02	0.37	0.19	0.05	0.07	0.01	0.03	0.05	0.10	0.16	0.07	0.08	0.16	0.05
N4	0.01	0.28	0.09	0.04	0.08	0.00	0.06	0.09	0.07	0.21	0.05	0.07	0.11	0.02
T1	0.29	0.78	0.41	0.13	0.25	0.38	0.44	0.34	0.36	0.25	0.13	0.33	0.32	0.20
LSC	0.69	1.00	0.94	0.71	0.66	0.47	0.94	0.47	0.83	0.91	0.72	0.90	0.83	0.76
C1	0.01	0.04	0.27	0.11	0.21	0.17	0.40	0.52	0.49	0.37	0.00	0.56	0.37	0.58
C2	0.04	0.09	0.45	0.22	0.38	0.33	0.59	0.65	0.65	0.54	0.00	0.71	0.53	0.65
Overall Complexity Score	**22.2**	**47.9**	**45.8**	**15.7**	**24.1**	**24.0**	**39.0**	**31.8**	**37.4**	**30.2**	**19.4**	**39.3**	**29.9**	**32.7**

Table 12. Data complexity analysis of the multiclass datasets augmented using the KFCGAN method across various measures.

Complexity Measures	**Dataset**													
	win	**cmc**	**bal**	**dry**	**gla**	**zoo**	**car**	**lym**	**wqr**	**yes**	**abs**	**wqw**	**aba**	**nur**
Density	0.80	0.86	0.76	0.62	0.81	0.78	0.71	0.80	0.76	0.74	0.75	0.73	0.63	0.82
ClsCoef	0.16	0.27	0.21	0.11	0.23	0.23	0.17	0.20	0.18	0.16	0.11	0.20	0.09	0.31
Hobs	0.68	0.72	0.67	0.58	0.70	0.66	0.59	0.68	0.62	0.62	0.62	0.60	0.53	0.69
L1	0.00	0.00	0.06	0.00	0.00	0.01	0.01	0.01	0.01	0.01	0.00	0.00	0.00	0.00
L2	0.01	0.02	0.15	0.00	0.01	0.07	0.05	0.06	0.06	0.05	0.02	0.05	0.03	0.04
L3	0.00	0.01	0.12	0.00	0.01	0.03	0.03	0.04	0.05	0.03	0.01	0.03	0.02	0.03
T2	0.09	0.01	0.01	0.00	0.06	0.21	0.00	0.11	0.01	0.01	0.05	0.00	0.01	0.00
T3	0.02	0.01	0.00	0.00	0.02	0.05	0.00	0.04	0.00	0.00	0.01	0.00	0.00	0.00
T4	0.23	0.70	0.58	0.10	0.35	0.23	0.56	0.40	0.35	0.35	0.19	0.50	0.25	0.53
F1	0.10	0.27	0.35	0.03	0.11	0.18	0.14	0.40	0.18	0.16	0.15	0.15	0.19	0.13
F1. v	0.01	0.06	0.21	0.01	0.02	0.06	0.07	0.10	0.10	0.07	0.03	0.08	0.05	0.08

Table 12 continued…

F2	0.03	0.01	0.07	0.00	0.00	0.00	0.00	0.00	0.04	0.01	0.00	0.00	0.05	0.01
F3	0.00	0.62	0.68	0.04	0.05	0.27	0.30	0.44	0.48	0.36	0.26	0.33	0.50	0.34
F4	0.00	0.09	0.63	0.01	0.00	0.01	0.12	0.07	0.29	0.18	0.07	0.12	0.27	0.18
N1	0.00	0.00	0.08	0.00	0.01	0.06	0.03	0.04	0.04	0.02	0.02	0.03	0.02	0.07
N2	0.07	0.13	0.11	0.03	0.10	0.19	0.08	0.21	0.12	0.10	0.08	0.08	0.08	0.13
N3	0.00	0.00	0.16	0.00	0.00	0.10	0.07	0.08	0.07	0.04	0.03	0.07	0.03	0.13
N4	0.00	0.01	0.15	0.00	0.01	0.07	0.06	0.07	0.05	0.04	0.03	0.05	0.03	0.11
T1	0.05	0.03	0.29	0.00	0.05	0.22	0.11	0.20	0.13	0.10	0.06	0.11	0.06	0.23
LSC	0.62	0.83	0.88	0.55	0.62	0.72	0.72	0.78	0.72	0.70	0.67	0.73	0.65	0.78
C1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
C2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Overall Complexity Score	**13.1**	**21.0**	**28.1**	**9.5**	**14.3**	**18.8**	**17.4**	**21.5**	**19.4**	**17.1**	**14.3**	**17.7**	**15.8**	**21.0**

Table 13. Data complexity analysis of the multiclass datasets resampled using the OCH-SMOTE method across various measures.

Complexity measures	**Dataset**													
	win	**cmc**	**bal**	**dry**	**gla**	**zoo**	**car**	**lym**	**wqr**	**yes**	**abs**	**wqw**	**aba**	**nur**
Density	0.76	0.72	0.94	0.73	0.86	0.81	0.95	0.82	0.88	0.83	0.85	0.85	0.74	0.88
ClsCoef	0.28	0.21	0.46	0.16	0.26	0.26	0.45	0.40	0.23	0.22	0.16	0.20	0.15	0.47
Hobs	0.66	0.64	0.82	0.61	0.73	0.68	0.84	0.68	0.74	0.70	0.73	0.70	0.61	0.71
L1	0.00	0.24	0.02	0.00	0.01	0.00	0.08	0.00	0.09	0.04	0.01	0.09	0.02	0.02
L2	0.01	0.34	0.06	0.02	0.06	0.00	0.18	0.02	0.16	0.08	0.02	0.19	0.09	0.05
L3	0.00	0.31	0.03	0.01	0.05	0.00	0.15	0.01	0.13	0.06	0.02	0.17	0.07	0.04
T2	0.09	0.01	0.01	0.00	0.06	0.21	0.00	0.11	0.01	0.01	0.05	0.00	0.01	0.00
T3	0.01	0.00	0.01	0.00	0.03	0.08	0.00	0.06	0.01	0.01	0.02	0.00	0.00	0.00
T4	0.08	0.37	1.00	0.29	0.47	0.40	1.00	0.57	0.74	0.70	0.45	0.78	0.41	0.91
F1	0.20	0.89	0.81	0.26	0.37	0.02	0.55	0.13	0.65	0.39	0.29	0.72	0.40	0.29
F1. v	0.03	0.64	0.17	0.05	0.09	0.03	0.35	0.04	0.33	0.20	0.13	0.40	0.20	0.15
F2	0.00	0.04	0.49	0.00	0.00	0.00	0.05	0.00	0.00	0.00	0.00	0.01	0.02	0.02
F3	0.22	0.95	0.97	0.52	0.43	0.03	0.67	0.17	0.80	0.56	0.32	0.82	0.48	0.41
F4	0.00	0.89	0.94	0.31	0.14	0.00	0.38	0.13	0.53	0.21	0.10	0.54	0.35	0.17
N1	0.08	0.18	0.07	0.01	0.03	0.01	0.04	0.02	0.07	0.04	0.09	0.06	0.13	0.03
N2	0.14	0.41	0.31	0.18	0.14	0.10	0.21	0.18	0.14	0.15	0.10	0.13	0.08	0.19
N3	0.12	0.34	0.11	0.02	0.03	0.00	0.03	0.03	0.05	0.04	0.02	0.03	0.03	0.04
N4	0.10	0.29	0.06	0.02	0.05	0.00	0.07	0.02	0.07	0.04	0.02	0.08	0.06	0.03
T1	0.31	0.76	0.34	0.10	0.18	0.15	0.21	0.18	0.21	0.16	0.11	0.18	0.10	0.18
LSC	0.86	1.00	0.96	0.87	0.79	0.55	0.93	0.60	0.93	0.83	0.74	0.94	0.75	0.84
C1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
C2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Overall Complexity Score	**17.9**	**41.9**	**38.9**	**18.9**	**21.7**	**15.2**	**32.4**	**18.9**	**30.8**	**23.9**	**19.1**	**31.3**	**21.4**	**24.7**

Table 14. Data complexity analysis of the multiclass datasets resampled using the SOMM method across various measures.

Complexity Measures	**Dataset**													
	win	**cmc**	**bal**	**dry**	**gla**	**zoo**	**car**	**lym**	**wqr**	**yes**	**abs**	**wqw**	**aba**	**nur**
Density	0.93	0.96	0.94	0.89	0.89	0.87	0.95	0.86	0.97	0.87	0.86	0.95	0.86	0.88
ClsCoef	0.59	0.44	0.50	0.49	0.50	0.51	0.52	0.50	0.48	0.51	0.32	0.56	0.32	0.49
Hobs	0.81	0.88	0.77	0.74	0.75	0.74	0.84	0.71	0.89	0.74	0.72	0.85	0.72	0.72
L1	0.00	0.14	0.02	0.50	0.01	0.00	0.06	0.00	0.01	0.00	0.10	0.01	0.10	0.02

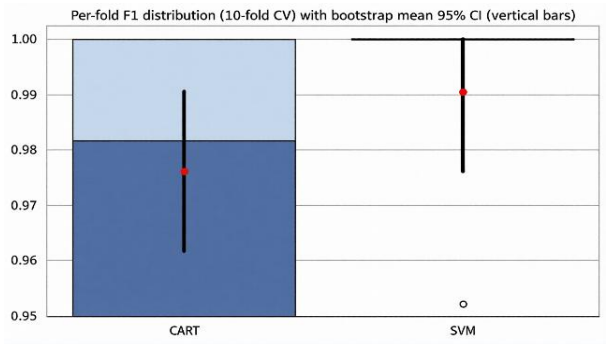
Table 14 continued…

L2	0.00	0.33	0.09	0.07	0.05	0.00	0.18	0.02	0.09	0.00	0.05	0.09	0.05	0.06
L3	0.00	0.30	0.06	0.05	0.04	0.00	0.15	0.01	0.06	0.00	0.03	0.06	0.03	0.04
T2	0.09	0.01	0.01	0.00	0.06	0.21	0.00	0.11	0.01	0.21	0.05	0.00	0.05	0.00
T3	0.01	0.00	0.01	0.00	0.03	0.01	0.00	0.05	0.00	0.01	0.00	0.00	0.00	0.00
T4	0.08	0.22	1.00	0.12	0.55	0.06	0.92	0.43	0.15	0.06	0.05	0.18	0.05	0.82
F1	0.16	0.86	0.80	0.16	0.29	0.02	0.56	0.16	0.39	0.02	0.25	0.37	0.25	0.28
F1. v	0.03	0.56	0.18	0.04	0.09	0.03	0.36	0.04	0.13	0.03	0.06	0.15	0.06	0.17
F2	0.00	0.68	1.00	0.00	0.00	0.00	0.11	0.00	0.02	0.00	0.00	0.02	0.00	0.10
F3	0.16	0.96	1.00	0.26	0.33	0.03	0.65	0.17	0.70	0.03	0.40	0.70	0.40	0.41
F4	0.00	0.96	1.00	0.12	0.06	0.00	0.39	0.14	0.40	0.00	0.10	0.38	0.10	0.17
N1	0.07	0.18	0.11	0.04	0.03	0.08	0.03	0.02	0.09	0.08	0.07	0.09	0.07	0.02
N2	0.13	0.42	0.38	0.06	0.26	0.23	0.29	0.24	0.22	0.23	0.05	0.19	0,05	0.26
N3	0.13	0.35	0.23	0.07	0.05	0.08	0.06	0.03	0.15	0.08	0.13	0.14	0.13	0.04
N4	0.10	0.29	0.08	0.07	0.04	0.16	0.08	0.02	0.15	0.16	0.17	0.16	0.17	0.03
T1	0.30	0.80	0.43	0.14	0.20	0.45	0.24	0.21	0.45	0.45	0.22	0.35	0.22	0.24
LSC	0.84	1.00	0.97	0.77	0.79	0.92	0.95	0.67	0.98	0.92	0.80	0.94	0.80	0.86
C1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
C2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Overall Complexity Score	**20.1**	**47.0**	**43.5**	**21.0**	**22.8**	**20.0**	**33.4**	**20.0**	**28.9**	**20.0**	**20.2**	**28.1**	**20.2**	**25.6**

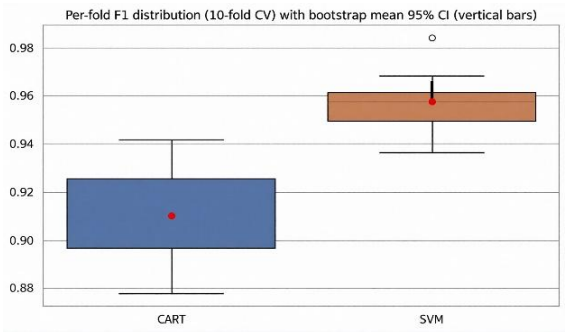
Table 15. Data complexity analysis of the multiclass datasets resampled using the SMOTE-CLS method across various measures.

Complexity measures	**Dataset**													
	win	**cmc**	**bal**	**dry**	**gla**	**zoo**	**car**	**lym**	**wqr**	**yes**	**abs**	**wqw**	**aba**	**nur**
Density	0.99	0.82	0.95	0.75	0.82	0.81	0.96	0.87	0.94	0.95	0.84	0.97	0.77	0.89
ClsCoef	0.75	0.25	0.47	0.23	0.25	0.14	0.48	0.32	0.34	0.21	0.07	0.23	0.15	0.47
Hobs	0.94	0.69	0.87	0.65	0.69	0.70	0.86	0.76	0.83	0.87	0.73	0.91	0.64	0.74
L1	0.00	0.00	0.01	0.00	0.00	0.00	0.09	0.00	0.03	0.03	0.01	0.05	0.01	0.02
L2	0.00	0.03	0.05	0.01	0.03	0.00	0.21	0.01	0.11	0.09	0.03	0.17	0.08	0.06
L3	0.00	0.02	0.03	0.00	0.02	0.00	0.18	0.00	0.08	0.06	0.02	0.14	0.07	0.04
T2	0.09	0.06	0.01	0.00	0.06	0.21	0.00	0.11	0.01	0.01	0.05	0.00	0.01	0.00
T3	0.07	0.04	0.01	0.00	0.04	0.08	0.00	0.06	0.01	0.01	0.02	0.00	0.00	0.00
T4	0.74	0.56	1.00	0.23	0.58	0.39	0.97	0.56	0.71	0.65	0.46	0.74	0.33	0.91
F1	0.14	0.30	0.80	0.09	0.30	0.02	0.60	0.13	0.48	0.31	0.34	0.64	0.35	0.30
F1. v	0.03	0.07	0.16	0.02	0.07	0.03	0.38	0.06	0.18	0.15	0.16	0.31	0.16	0.16
F2	0.00	0.00	1.00	0.00	0.00	0.00	0.11	0.04	0.02	0.01	0.01	0.03	0.03	0.10
F3	0.16	0.34	1.00	0.26	0.34	0.05	0.72	0.20	0.77	0.54	0.44	0.82	0.42	0.43
F4	0.00	0.07	1.00	0.17	0.07	0.00	0.40	0.15	0.49	0.28	0.14	0.47	0.27	0.17
N1	0.01	0.03	0.07	0.00	0.03	0.01	0.00	0.08	0.03	0.08	0.26	0.18	0.20	0.11
N2	0.26	0.17	0.33	0.09	0.17	0.03	0.17	0.14	0.16	0.06	0.03	0.06	0.03	0.15
N3	0.02	0.04	0.11	0.01	0.04	0.00	0.00	0.02	0.03	0.01	0.01	0.01	0.01	0.03
N4	0.00	0.05	0.06	0.00	0.05	0.00	0.09	0.01	0.07	0.05	0.03	0.12	0.08	0.05
T1	0.25	0.18	0.33	0.04	0.18	0.11	0.16	0.15	0.20	0.09	0.07	0.10	0.05	0.15
LSC	0.68	0.76	0.96	0.74	0.76	0.57	0.95	0.64	0.91	0.82	0.76	0.95	0.74	0.88
C1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
C2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Overall Complexity Score	**23.4**	**20.4**	**41.9**	**15.0**	**20.4**	**14.4**	**33.3**	**19.6**	**29.1**	**24.0**	**20.3**	**31.3**	**20.0**	**25.7**

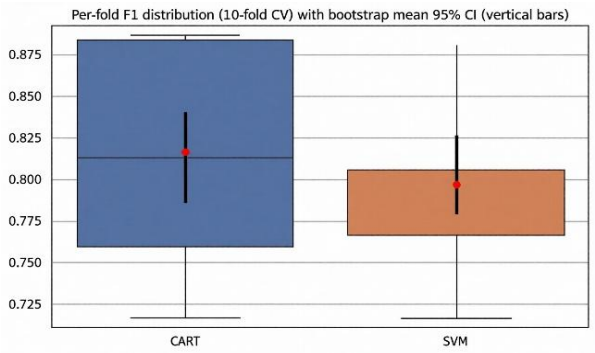
A.3: Additional results of the wilcoxon signed-rank test using 10-fold cross-validation and 95% bootstrap confidence intervals.



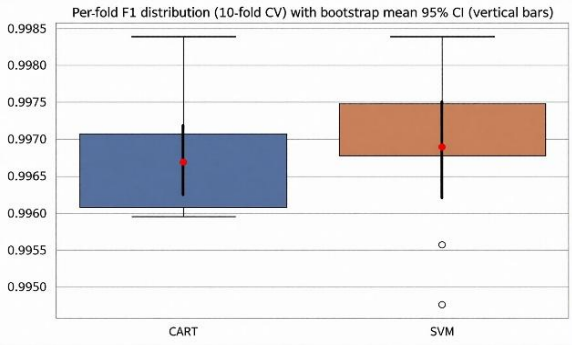
(a)



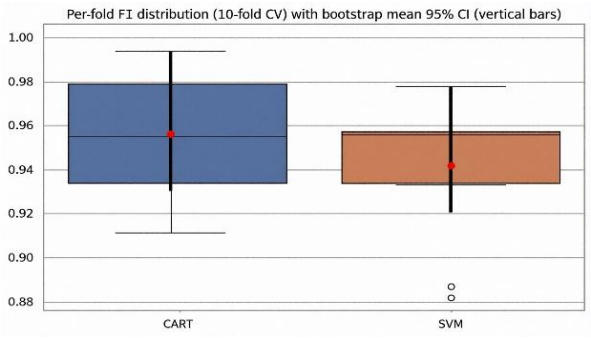
(b)



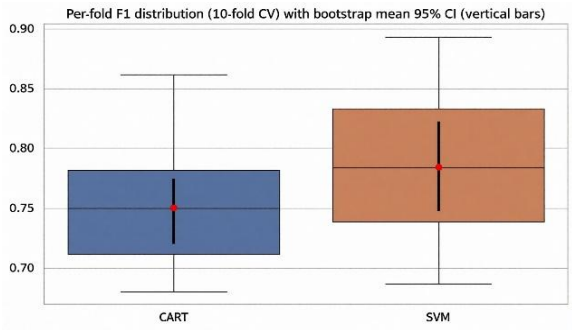
(c)



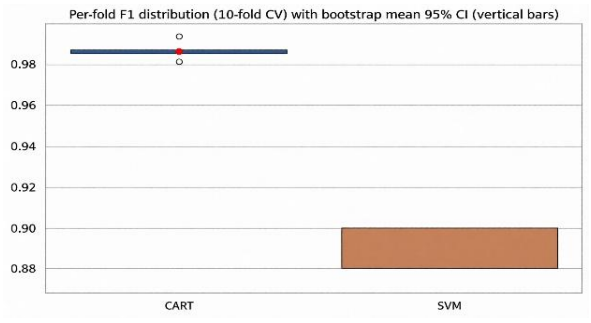
(d)



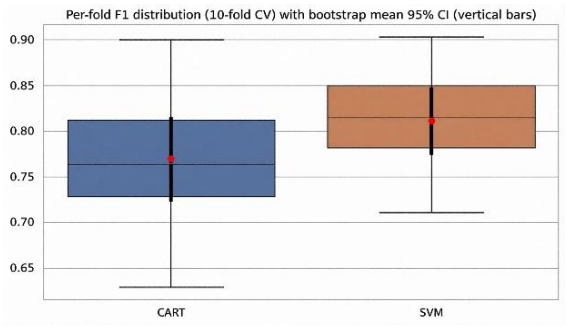
(e)



(f)



(g)



(h)

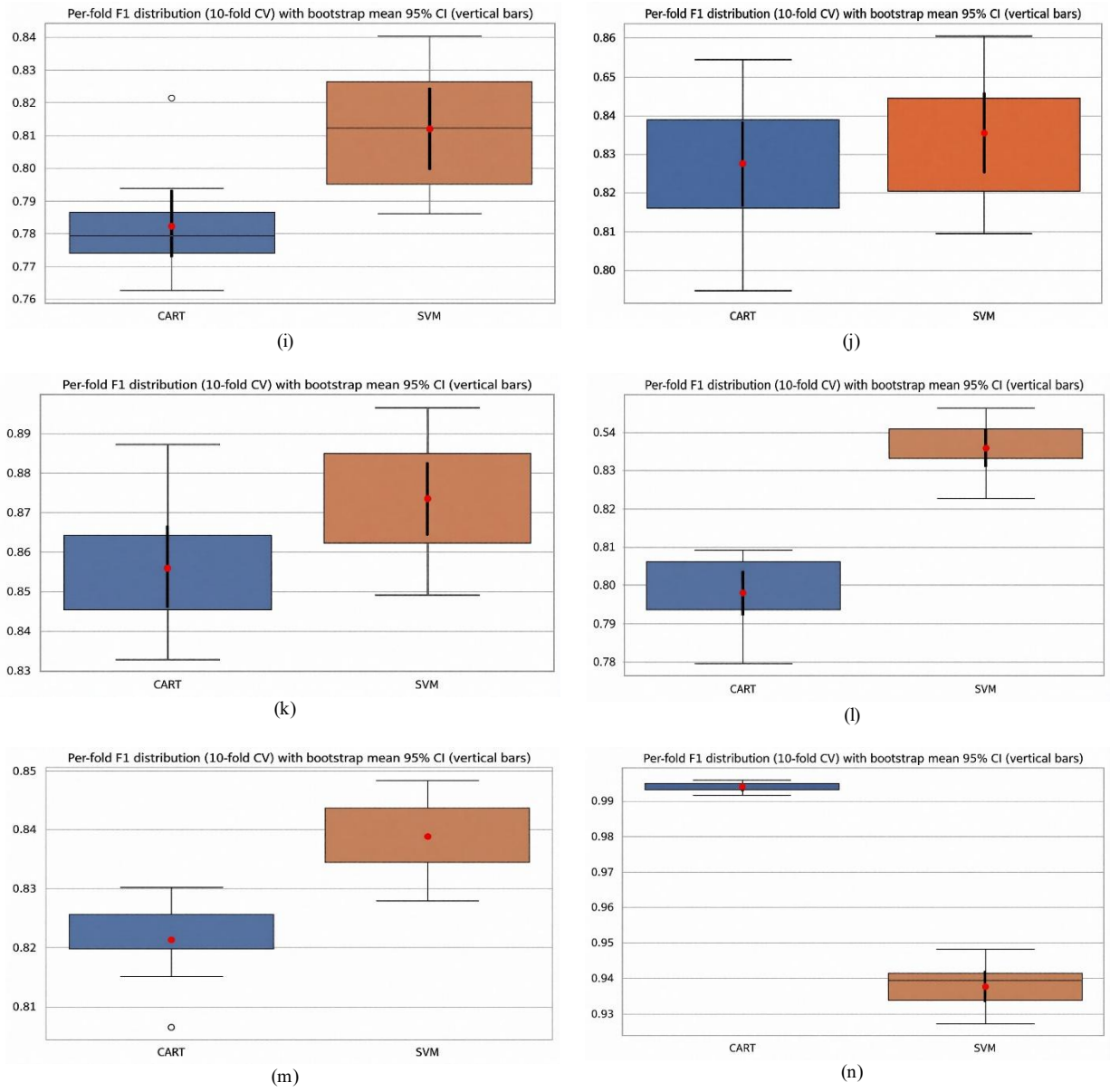
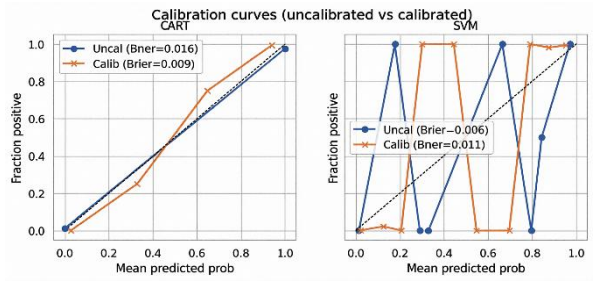
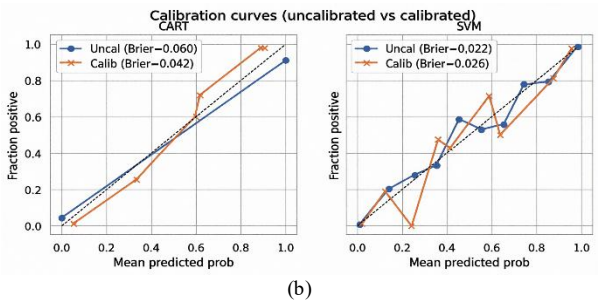


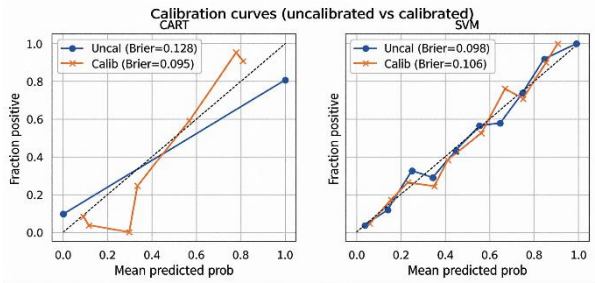
Figure 5. Per-fold F1-score distribution (10-fold cross-validation) comparing CART and SVM across the datasets: (a) win, (b) cmc, (c) bal, (d) dry, (e) gla, (f) zoo, (g) car, (h) lym, (i) wqr, (j) yes, (k) abs, (l) wqw, (m) aba, and (n) nur.


Calibration curves (uncalibrated vs calibrated)
CART
Uncal (Bner=0.016)
Calib (Brier=0.009)
SVM
Uncal (Brier=0.006)
Calib (Bner=0.011)
Fraction positive
Mean predicted prob

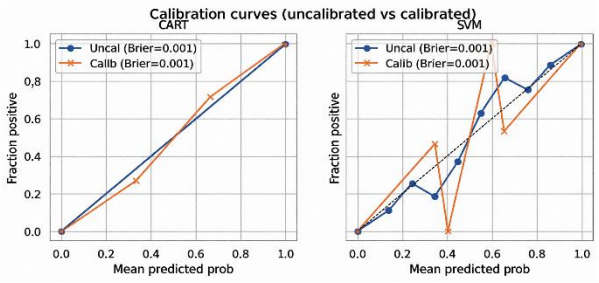
(a)


Calibration curves (uncalibrated vs calibrated)
CART
Uncal (Brier=0.060)
Calib (Brier=0.042)
SVM
Uncal (Brier=0.022)
Calib (Brier=0.026)
Fraction positive
Mean predicted prob

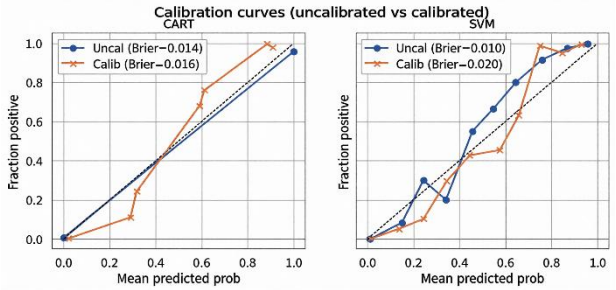
(b)


Calibration curves (uncalibrated vs calibrated)
CART
Uncal (Brier=0.128)
Calib (Brier=0.095)
SVM
Uncal (Brier=0.098)
Calib (Brier=0.106)
Fraction positive
Mean predicted prob

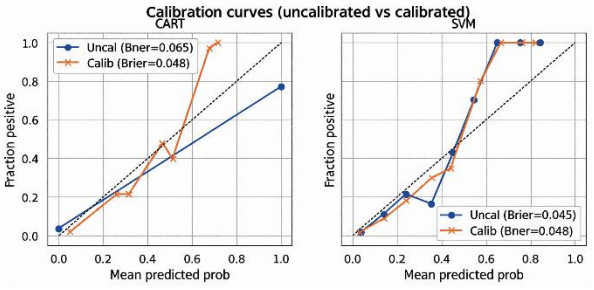
(c)


Calibration curves (uncalibrated vs calibrated)
CART
Uncal (Brier=0.001)
Calib (Brier=0.001)
SVM
Uncal (Brier=0.001)
Calib (Brier=0.001)
Fraction positive
Mean predicted prob

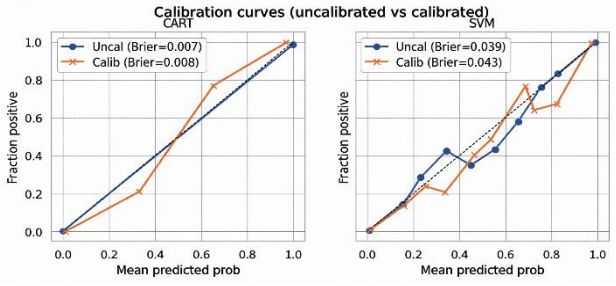
(d)


Calibration curves (uncalibrated vs calibrated)
CART
Uncal (Brier=0.014)
Calib (Brier=0.016)
SVM
Uncal (Brier=0.010)
Calib (Brier=0.020)
Fraction positive
Mean predicted prob

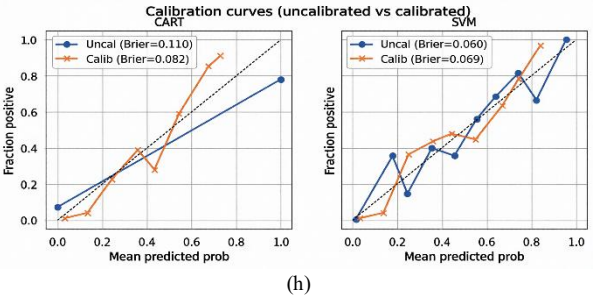
(e)


Calibration curves (uncalibrated vs calibrated)
CART
Uncal (Bner=0.065)
Calib (Brier=0.048)
SVM
Uncal (Brier=0.045)
Calib (Bner=0.048)
Fraction positive
Mean predicted prob

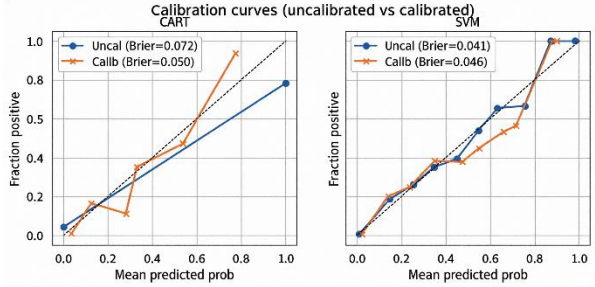
(f)


Calibration curves (uncalibrated vs calibrated)
CART
Uncal (Brier=0.007)
Calib (Brier=0.008)
SVM
Uncal (Brier=0.039)
Calib (Brier=0.043)
Fraction positive
Mean predicted prob

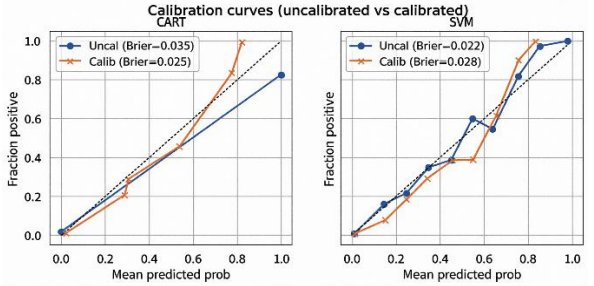
(g)


Calibration curves (uncalibrated vs calibrated)
CART
Uncal (Brier=0.110)
Calib (Brier=0.082)
SVM
Uncal (Brier=0.060)
Calib (Brier=0.069)
Fraction positive
Mean predicted prob

(h)


Calibration curves (uncalibrated vs calibrated)
CART
Uncal (Brier=0.072)
Calib (Brier=0.050)
SVM
Uncal (Brier=0.041)
Calib (Brier=0.046)
Fraction positive
Mean predicted prob

(i)


Calibration curves (uncalibrated vs calibrated)
CART
Uncal (Brier=0.035)
Calib (Brier=0.025)
SVM
Uncal (Brier=0.022)
Calib (Brier=0.028)
Fraction positive
Mean predicted prob

(j)

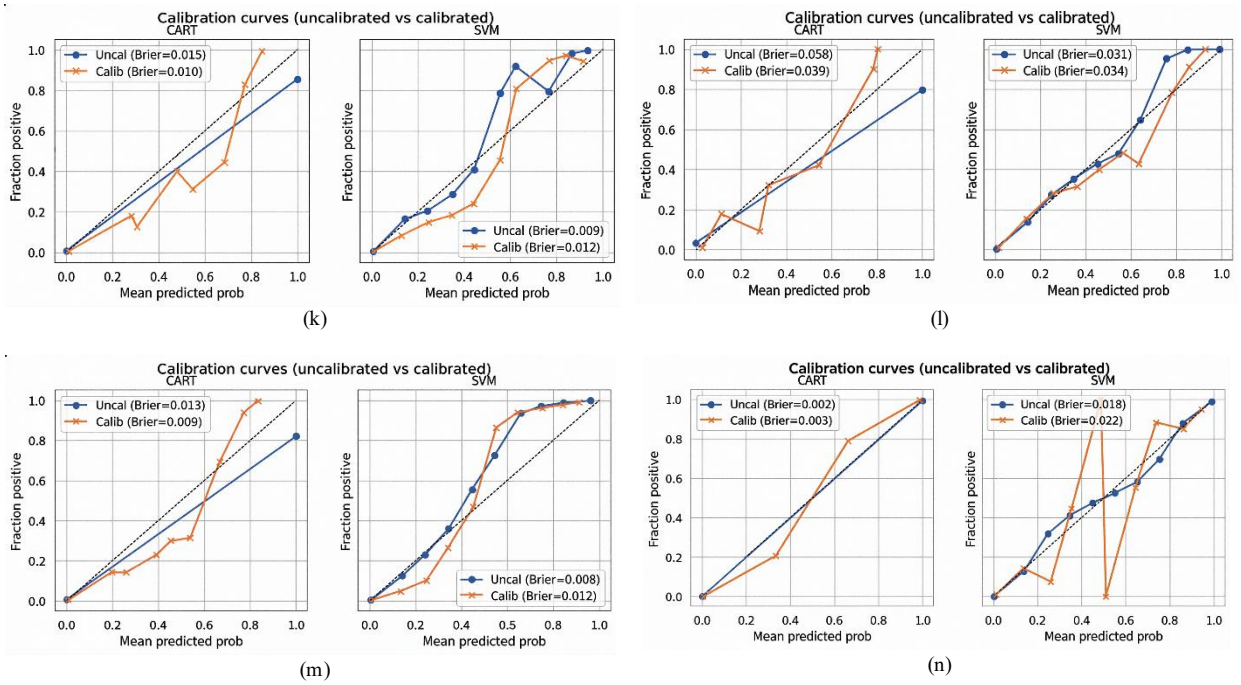


Figure 6. Calibration curves from 10-fold cross-validation comparing the probabilistic reliability of CART and SVM across the datasets: (a) win, (b) cmc, (c) bal, (d) dry, (e) gla, (f) zoo, (g) car, (h) lym, (i) wqr, (j) yes, (k) abs, (l) wqw, (m) aba, and (n) nur.

A.4: Limitations

CGAN and OVO have higher computing costs but for different reasons. CGAN needs to train a generator and a discriminator simultaneously in an iterative adversarial learning process with many epochs of forward and backward propagations, which makes the generation of synthetic samples much slower than the OCH-SMOTE, SOMM, and SMOTE-CLS. On the other hand, OVO increases the computational complexity because it breaks down a multiclass problem with C classes into C(C-1)/2 binary classification tasks, necessitating the training and evaluation of a large number of classifiers. As the number of courses increases, training and prediction times also increase significantly. Therefore, although CGAN and OVO can effectively handle multiclass imbalance situations, their computing needs become higher for large-scale datasets with multiple classes and high-dimensional feature spaces. When the dataset has too few samples to determine the data's true distribution, CGAN training may become unstable.

Conflicts of Interest

The authors confirm that there is no conflict of interest to declare for this publication.

Acknowledgments

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors. The authors would like to thank the editor and anonymous reviewers for their comments that help improve the quality of this work.

AI Disclosure

The authors declare that no assistance is taken from generative AI to write this article.

References

Abdi, L., & Hashemi, S. (2016). To combat multi-class imbalanced problems by means of over-sampling techniques. *IEEE Transactions on Knowledge and Data Engineering*, *28*(1), 238-251. https://doi.org/10.1109/TKDE.2015.2458858

Ahsan, R., Shi, W., Ma, X., & Croft, W.L. (2022). A comparative analysis of CGAN-based oversampling for anomaly detection. *IET Cyber-Physical Systems: Theory and Applications*, *7*(1), 40-50. https://doi.org/10.1049/cps2.12019

Alabduallah, B., Maray, M., Alruwais, N., Alabdan, R., Darem, A.A., Alallah, F.S., Alsini, R., & Yafoz, A. (2024). Class imbalanced data handling with cyberattack classification using hybrid Salp swarm algorithm with deep learning approach. *Alexandria Engineering Journal*, *106*, 654-663. https://doi.org/10.1016/j.aej.2024.08.061

Altalhan, M., Algarni, A., & Alouane, M.T.H. (2025). Imbalanced data problem in machine learning: a review. *IEEE Access*, *13*, 13686-13699. https://doi.org/10.1109/ACCESS.2025.3531662

Asmat, B.N., Bilal, H.S.M., Uddin, M.I., Karim, F.K., & Mostafa, S.M. (2025). Utilizing conditional GANs for synthesis of equilibrated hyperspectral data to enhance classification and mitigate majority class bias. *IEEE Access*, *13*, 49271-49289. https://doi.org/10.1109/ACCESS.2025.3550057

Bae, W.D., Alkobaisi, S., Horak, M., Bankar, S., Bhuvaji, S., Kim, S., & Park, C.S. (2025). Synthetic data generation and evaluation techniques for classifiers in data starved medical applications. *IEEE Access*, *13*, 16584-16602. https://doi.org/10.1109/ACCESS.2025.3532222

Batuwita, R., & Palade, V. (2013). Class imbalance learning methods for support vector machines. In: He, H., & Ma, Y. (eds) *Imbalanced Learning: Foundations, Algorithms, and Applications* (pp. 83-89). Wiley Online Library. https://doi.org/10.1002/9781118646106.ch5

Bayram, F., Ahmed, B.S., & Kassler, A. (2022). From concept drift to model degradation: an overview on performance-aware drift detectors. *Knowledge-Based Systems*, *245*, 108632. https://doi.org/10.1016/j.knosys.2022.108632

Brishti, F., Zhang, F., Mohammed, S., Bai, L., Wu, F., & Chen, B. (2025). Imbalanced classification with label noise: a systematic review and comparative analysis. *ICT Express*, *11*(6), 1127-1145. https://doi.org/10.1016/j.icte.2025.09.011

Cemernek, D., Siddiqi, S., & Kern, R. (2024). Effects of class imbalance countermeasures on interpretability. *IEEE Access*, *12*, 45342-45358. https://doi.org/10.1109/ACCESS.2024.3381536

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.

Cohen, J. (1960). A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, *20*(1), 37-46. https://doi.org/10.1177/001316446002000104

Collell, G., Prelec, D., & Patil, K.R. (2018). A simple plug-in bagging ensemble based on threshold-moving for classifying binary and multiclass imbalanced data. *Neurocomputing*, *275*, 330-340. https://doi.org/10.1016/j.neucom.2017.08.035

Creswell, A., White, T., Dumoulin, V., Arulkumaran, K., Sengupta, B., & Bharath, A.A. (2018). Generative adversarial networks: an overview. *IEEE Signal Processing Magazine*, *35*(1), 53-65. https://doi.org/10.1109/MSP.2017.2765202

Cusworth, S., Gkoutos, G.V., & Acharjee, A. (2024). A novel generative adversarial networks modelling for the class imbalance problem in high dimensional omics data. *BMC Medical Informatics and Decision Making*, *24*(1), 90. https://doi.org/10.1186/s12911-024-02487-2

Dikananda, A.R., Jumini, S., Tarihoran, N., Christinawati, S., Trimastuti, W., & Rahim, R. (2022). Comparison of decision tree classification methods and gradient boosted trees. *TEM Journal*, *11*(1), 316-322. https://doi.org/10.18421/TEM111-39

Eberlein, J., Rodriguez, D., & Harrison, R. (2025). The effect of data complexity on classifier performance. *Empirical Software Engineering*, *30*(1), 16. https://doi.org/10.1007/s10664-024-10554-5

Fernández, A., Garcia, S., Herrera, F., & Chawla, N.V. (2018). SMOTE for learning from imbalanced data: progress and challenges, marking the 15-year anniversary. *Journal of Artificial Intelligence Research*, *61*, 863-905.

Galar, M., Fernández, A., Barrenechea, E., Bustince, H., & Herrera, F. (2011). An overview of ensemble methods for binary classifiers in multi-class problems: experimental study on one-vs-one and one-vs-all schemes. *Pattern Recognition*, *44*(8), 1761-1776. https://doi.org/10.1016/j.patcog.2011.01.017

Ganie, S.M., & Pramanik, P.K.D. (2024). A comparative analysis of boosting algorithms for chronic liver disease prediction. *Healthcare Analytics*, *5*, 100313. https://doi.org/10.1016/j.health.2024.100313

Garcia-Pedrajas, N., Cuevas-Munoz, J.M., & de Haro-Garcia, A. (2024). Evolutionary simultaneous under and oversampling of instances for dealing with class-imbalance datasets in multilabel problems. *Applied Soft Computing*, *159*, 111618. https://doi.org/10.1016/j.asoc.2024.111618

Geetha, G., & Geethanjali, P. (2024). An efficient method for bearing fault diagnosis. *Systems Science & Control Engineering*, *12*(1), 2329264. https://doi.org/10.1080/21642583.2024.2329264

Ghorbani, H. (2019). Mahalanobis distance and its application for detecting multivariate outliers. *Facta Universitatis, Series: Mathematics and Informatics*, *34*(3), 583-595. https://doi.org/10.22190/fumi1903583g

Goodfellow, I.J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2014). *Generative Adversarial Networks*. http://arxiv.org/abs/1406.2661

Guan, C., Shang, R., Yang, F., Shao, A.X., & Zhang, S. (2024). A priori information-guided generative adversarial network for data augmentation: application to pipeline fault diagnosis. *Systems Science & Control Engineering*, *12*(1), 2343311. https://doi.org/10.1080/21642583.2024.2343311

Hartono, & Syah, R.B.Y. (2024). Hybrid approach with membership-density based oversampling for handling multi-class imbalance in internet traffic identification with overlapping and noise. *ICT Express*, *10*(5), 1094-1102. https://doi.org/10.1016/j.icte.2024.04.007

Haykin, S. (2001). Kalman filters. In: Haykin, S. (ed) *Kalman Filtering and Neural Networks* (pp. 1-21). Wiley Online Library. https://doi.org/10.1002/0471221546.ch1

Hirsch, V., Reimann, P., Treder-Tschechlov, D., Schwarz, H., & Mitschang, B. (2023). Exploiting domain knowledge to address class imbalance and a heterogeneous feature space in multi-class classification. The *VLDB Journal*, *32*(5), 1037-1064. https://doi.org/10.1007/s00778-023-00780-6

Hong, S., An, S., & Jeon, J.J. (2025). Improving SMOTE via fusing conditional VAE for data-adaptive noise filtering. *Applied Intelligence*, *55*(12), 841. https://doi.org/10.1007/s10489-025-06692-y

Hüllermeier, E., & Vanderlooy, S. (2010). Combining predictions in pairwise classification: an optimal adaptive voting strategy and its relation to weighted voting. *Pattern Recognition*, *43*(1), 128-142. https://doi.org/10.1016/j.patcog.2009.06.013

Kalman, R.E. (1960). A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, *82*(1), 35-45. https://doi.org/10.1115/1.3662552

Kannan, M., Umamaheswari, D., Manimekala, B., Mary, I.P.S., Savitha, P.M., & Rozario, J. (2025). An enhancement of machine learning model performance in disease prediction with synthetic data generation. *Scientific Reports*, *15*(1), 33482. https://doi.org/10.1038/s41598-025-15019-3

Khorshidi, H.A., & Aickelin, U. (2025). A synthetic over-sampling method with minority and majority classes for imbalance problems. *Knowledge and Information Systems*, *67*(7), 5965-5998. https://doi.org/10.1007/s10115-025-02394-6

Komorniczak, J., & Ksieniewicz, P. (2022). Problexity -- an open-source Python library for binary classification problem complexity assessment. http://arxiv.org/abs/2207.06709

Koziarski, M. (2020). Radial-based undersampling for imbalanced data classification. *Pattern Recognition*, *102*, 107262. https://doi.org/10.1016/j.patcog.2020.107262

Koziarski, M., Krawczyk, B., & Woźniak, M. (2019). Radial-based oversampling for noisy imbalanced data classification. *Neurocomputing*, *343*, 19-33. https://doi.org/10.1016/j.neucom.2018.04.089

Koziarski, M., Woźniak, M., & Krawczyk, B. (2020). Combined cleaning and resampling algorithm for multi-class imbalanced data with label noise. http://arxiv.org/abs/2004.03406

Kubat, M. (1997). Addressing the curse of imbalanced training sets: one-sided selection. In *Proceedings of the 14th International Conference on Machine Learning* (pp. 179-186). Morgan Kaufmann. San Francisco, CA, United States.

Kwon, H., Greenberg, M., Josephson, C.B., & Lee, J. (2024). Measuring the prediction difficulty of individual cases in a dataset using machine learning. *Scientific Reports*, *14*(1), 10474. https://doi.org/10.1038/s41598-024-61284-z

Lancho, C., De Diego, I.M., Cuesta, M., Aceña, V., & Moguerza, J.M. (2023). Hostility measure for multi-level study of data complexity. *Applied Intelligence*, *53*(7), 8073-8096. https://doi.org/10.1007/s10489-022-03793-w

Lin, C.L., & Fan, C.L. (2019). Evaluation of CART, CHAID, and QUEST algorithms: a case study of construction defects in Taiwan. *Journal of Asian Architecture and Building Engineering*, *18*(6), 539-553. https://doi.org/10.1080/13467581.2019.1696203

Liu, W., Zhang, H., Ding, Z., Liu, Q., & Zhu, C. (2021). A comprehensive active learning method for multiclass imbalanced data streams with concept drift. *Knowledge-Based Systems*, *215*, 106778. https://doi.org/10.1016/j.knosys.2021.106778

Liu, W., Zhu, C., Ding, Z., Zhang, H., & Liu, Q. (2023). Multiclass imbalanced and concept drift network traffic classification framework based on online active learning. *Engineering Applications of Artificial Intelligence*, *117*, 105607. https://doi.org/10.1016/j.engappai.2022.105607

Ma, H., Yan, L., Xia, Y., & Fu, M. (2020). Introduction to Kalman filtering. In: Ma, H., Yan, L., Xia, Y., & Fu, M. (eds) *Kalman Filtering and Information Fusion*. Springer, Singapore. pp. 3-9. https://doi.org/10.1007/978-981-15-0806-6_1

Martínez, P.M.P., Forradellas, R.F.R., Gallastegui, L.M.G., & Alonso, S.L.N. (2025). Comparative analysis of machine learning models for the detection of fraudulent banking transactions. *Cogent Business & Management*, *12*(1), 2474209. https://doi.org/10.1080/23311975.2025.2474209

Mert, A. (2023). Enhanced dataset synthesis using conditional generative adversarial networks. *Biomedical Engineering Letters*, *13*(1), 41-48. https://doi.org/10.1007/s13534-022-00251-x

Min, G., & Oh, J. (2025). Can synthetic data protect privacy? *IEEE Access*, *13*, 31544-31561. https://doi.org/10.1109/ACCESS.2025.3542266

Mirza, M., & Osindero, S. (2014). Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*.

Naglik, I., & Lango, M. (2024). GMMSampling: a new model-based, data difficulty-driven resampling method for multi-class imbalanced data. *Machine Learning*, *113*(8), 5183-5202. https://doi.org/10.1007/s10994-023-06416-8

Olabisi, O., Maurya, L., & Bader-El-Den, M. (2025). Localised ensemble learning (LEL) – a localised approach to class imbalance. *Pattern Analysis and Applications*, *28*(4), 177. https://doi.org/10.1007/s10044-025-01545-3

Palli, A.S., Jaafar, J., Hashmani, M.A., Gomes, H.M., & Gilal, A.R. (2022). A hybrid sampling approach for imbalanced binary and multi-class data using clustering analysis. *IEEE Access*, *10*, 118639-118653. https://doi.org/10.1109/ACCESS.2022.3218463

Pan, Z., Yu, W., Yi, X., Khan, A., Yuan, F., & Zheng, Y. (2019). Recent progress on generative adversarial networks (GANs): a survey. *IEEE Access*, *7*, 36322-36333. https://doi.org/10.1109/ACCESS.2019.2905015

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, É. (2011). Scikit-learn: machine learning in Python. *The Journal of Machine Learning Research*, *12*, 2825-2830.

Qin, J., Yang, D., Zhou, B., & Sun, Y. (2024). Semi-supervised variational bi-directional sampling on multi-class imbalanced electric power data for fault diagnosis. *International Journal of Electrical Power & Energy Systems*, *155*, 109512. https://doi.org/10.1016/j.ijepes.2023.109512

Ramachandra, P., & Vaithiyanathan, S. (2025). Fed-DPSDG-WGAN: differentially private synthetic data generation for loan default prediction via federated wasserstein GAN. *IEEE Access*, *13*, 52069-52084. https://doi.org/10.1109/ACCESS.2025.3552487

Rawat, S.S., Thada, V., & Mishra, A.K. (2026). SMOTE-AAE-RF: a novel approach to identifying financial fraud in imbalanced data circumstances. *International Journal of Mathematical, Engineering and Management Sciences, 11*(2), 850. https://doi.org/10.33889/IJMEMS.2026.11.2.036

Rey, D., Neuhäuser, M. (2011). Wilcoxon-signed-rank test. In: Lovric, M. (eds) *International Encyclopedia of Statistical Science*. Springer, Berlin, Heidelberg. pp. 1658-1659. https://doi.org/10.1007/978-3-642-04898-2_616

Sadeghi, F., Viktor, H.L., & Vafaie, P. (2023). DynaQ: online learning from imbalanced multi-class streams through dynamic sampling. *Applied Intelligence*, *53*(21), 24908-24930. https://doi.org/10.1007/s10489-023-04886-w

Shana, T.B., Kumari, N., Agarwal, M., Mondal, S., & Rathnayake, U. (2025). Anomaly-based intrusion detection system based on SMOTE-IPF, whale optimization algorithm, and ensemble learning. *Intelligent Systems with Applications*, *27*, 200543. https://doi.org/10.1016/j.iswa.2025.200543

Sharma, P., Kumar, M., Sharma, H.K., & Biju, S.M. (2024). Generative adversarial networks (GANs): introduction, taxonomy, variants, limitations, and applications. *Multimedia Tools and Applications*, *83*(41), 88811-88858. https://doi.org/10.1007/s11042-024-18767-y

Sokolova, M., & Lapalme, G. (2009). A systematic analysis of performance measures for classification tasks. *Information Processing and Management*, *45*(4), 427-437. https://doi.org/10.1016/j.ipm.2009.03.002

Suwanda, R., Syahputra, Z., & Zamzami, E.M. (2020). Analysis of Euclidean distance and Manhattan distance in the K-means algorithm for variations number of centroid K. In *Journal of Physics: Conference Series* (Vol. 1566, No. 1, p. 012058). IOP Publishing. Medan, Indonesia. https://doi.org/10.1088/1742-6596/1566/1/012058

Udu, A.G., Salman, M.T., Ghalati, M.K., Lecchini-Visintini, A., Siddle, D.R., & Dong, H. (2025). Emerging SMOTE and GAN-variants for data augmentation in imbalance machine learning tasks: a review. *IEEE Access*, *13*, 113838-113853. https://doi.org/10.1109/ACCESS.2025.3584532

Wan, X., Zheng, Z., Qin, F., & Lu, X. (2023). Data complexity: a new perspective for analyzing the difficulty of defect prediction tasks. http://arxiv.org/abs/2305.03615

Wang, A.X., Le, M.Q., Duong, H.T., Van, B.N., & Nguyen, B.P. (2025). CTVAE: contrastive tabular variational autoencoder for imbalance data. *Knowledge and Information Systems*, *67*(6), 5335-5354. https://doi.org/10.1007/s10115-025-02377-7

Wang, J., & Awang, N. (2024). MKC-SMOTE: a novel synthetic oversampling method for multi-class imbalanced data classification. *IEEE Access*, *12*, 196929-196938. https://doi.org/10.1109/ACCESS.2024.3521120

Wang, J., & Awang, N. (2025). A novel synthetic minority oversampling technique for multiclass imbalance problems. *IEEE Access*, *13*, 6054-6066. https://doi.org/10.1109/ACCESS.2025.3526673

Wang, S., & Yao, X. (2012). Multiclass imbalance problems: analysis and potential solutions. *IEEE Transactions on Systems, Man, and Cybernetics, Part B* (*Cybernetics*), *42*(4), 1119-1130. https://doi.org/10.1109/TSMCB.2012.2187280

Wickramaratne, S.D., & Mahmud, M.S. (2021). Conditional-GAN based data augmentation for deep learning task classifier improvement using fNIRS data. *Frontiers in Big Data*, *4*, 659146. https://doi.org/10.3389/fdata.2021.659146

Xie, Z., & Huang, X. (2024). A credit card fraud detection method based on mahalanobis distance hybrid sampling and random forest algorithm. *IEEE Access*, *12*, 162788-162798. https://doi.org/10.1109/ACCESS.2024.3421316

Yan, Y., Lv, Y., Han, S., Yu, C., & Zhou, P. (2025). GDHS: an efficient hybrid sampling method for multi-class imbalanced data classification. *Neurocomputing*, *637*, 130088. https://doi.org/10.1016/j.neucom.2025.130088

Yu, J.R., Lin, C.Y., & Lien, D. (2025). Cost-effective and accuracy-oriented ℓ1-norm support vector machine for enhanced feature selection. *Measurement*, *253*, 117506. https://doi.org/10.1016/j.measurement.2025.117506

Zhang, S. (2021). Challenges in KNN classification. *IEEE Transactions on Knowledge and Data Engineering*, *34*(10), 4663-4675. https://doi.org/10.1109/TKDE.2021.3049250

Zhang, Z., Krawczyk, B., Garcia, S., Rosales-Pérez, A., & Herrera, F. (2016). Empowering one-vs-one decomposition with ensemble learning for multi-class imbalanced data. *Knowledge-Based Systems*, *106*, 251-263. https://doi.org/10.1016/j.knosys.2016.05.048

Zhang, Z.L., Luo, X.G., González, S., García, S., & Herrera, F. (2018). DRCW-ASEG: one-versus-one distance-based relative competence weighting with adaptive synthetic example generation for multi-class imbalanced datasets. *Neurocomputing*, *285*, 176-187. https://doi.org/10.1016/j.neucom.2018.01.039

Zhao, G., Liu, P., Sun, K., Yang, Y., Lan, T., & Yang, H. (2023). Research on data imbalance in intrusion detection using CGAN. *Plos one*, *18*(10), e0291750. https://doi.org/10.1371/journal.pone.0291750

Zhu, T., Lin, Y., & Liu, Y. (2017). Synthetic minority oversampling technique for multiclass imbalance problems. *Pattern Recognition*, *72*, 327-340. https://doi.org/10.1016/j.patcog.2017.07.024