Machine Learning
https://doi.org/10.1007/s10994-023-06448-0

Solving imbalanced learning with outlier detection and features reduction

Salvatore Lusito[1] · **Andrea Pugnana**[2] · **Riccardo Guidotti**[1,3]

Received: 21 June 2022 / Revised: 28 August 2023 / Accepted: 7 October 2023

Abstract
A critical problem for several real world applications is class imbalance. Indeed, in contexts like fraud detection or medical diagnostics, standard machine learning models fail because they are designed to handle balanced class distributions. Existing solutions typically increase the rare class instances by generating synthetic records to achieve a balanced class distribution. However, these procedures generate not plausible data and tend to create unnecessary noise. We propose a change of perspective where instead of relying on resampling techniques, we depend on unsupervised features engineering approaches to represent records with a combination of features that will help the classifier capturing the differences among classes, even in presence of imbalanced data. Thus, we combine a large array of outlier detection, features projection, and features selection approaches to augment the expressiveness of the dataset population. We show the effectiveness of our proposal in a deep and wide set of benchmarking experiments as well as in real case studies.

Keywords Imbalanced data learning · Outlier detection · Features reduction · Features selection · Classification framework

Editors: Nuno Moniz, Paula Branco, Luís Torgo, Nathalie Japkowicz, Michal Wozniak, Shuo Wang.

✉ Riccardo Guidotti
riccardo.guidotti@unipi.it

Salvatore Lusito
s.lusito@studenti.unipi.it

Andrea Pugnana
andrea.pugnana@sns.it

[1] University of Pisa, Largo B. Pontecorvo, 56127 Pisa, Italy

[2] Scuola Normale Superiore, Piazza dei Cavalieri, 7, Pisa 56127, Italy

[3] ISTI-CNR, Via G. Moruzzi, 56127 Pisa, Italy

Published online: 07 December 2023

1 Introduction

A wide variety of important applications where Machine Learning (ML) models are typically applied, such as fraud detection, medical diagnosis, and oil spill (Japkowicz & Stephen, 2002) suffer from the problem of class imbalance. The class imbalance problem corresponds to domains where some classes are represented by a large number of instances, while others are represented by only a few. Prior research has shown that class imbalance has a negative impact on the performance of the learned ML models that tend to be overwhelmed by the large classes and ignore the small ones. This typically happens because ML classifiers operate on data drawn from the same distribution as the training data, adopting the same data representation, and assume that maximizing accuracy is the primary goal (Chawla et al., 2004). Given the frequency of imbalanced learning problems in real applications and the issues it raises on learning ML models, the research of methods for handling it has become a significant research topic (Japkowicz & Stephen, 2002; Branco et al., 2016).

The approaches proposed by the research community to solve the class imbalance problem include pre-processing methods, as well as the definition of learning methods specifically designed for this problem (Japkowicz & Stephen, 2002; Chawla et al., 2004; He & Garcia, 2009; Chawla, 2010; Branco et al., 2016). Pre-processing and resampling approaches are more widely studied as they enable the subsequent adoption of any standard ML classification model. The main ideas consist in transforming the original training set, making it more suitable for learning the important class(es) either by reducing the number of instances belonging to the majority classes, or by augmenting the number of rare instances through synthetic data generation procedures (He & Garcia, 2009). Well known examples are the Random Undersampling (Kubat & Matwin, 1997) and the Condensed Nearest Neighbors (Hart, 1968) procedures for data reduction, or the Random Oversampling (Kubat & Matwin, 1997) and the Synthetic Minority Oversampling Technique (Chawla et al., 2002) for data augmentation. Many proposals in the literature try to refine such basic approaches by combining the aforementioned solutions in different fashions or by resorting to advanced Automated Machine Learning (He et al., 2021) approaches. Unfortunately, despite handling somewhat with the class imbalance problem, many of the most widely used pre-processing approaches suffer from issues related to the removal of majority class instances from sparse regions, and to the generation of noisy/erroneous synthetic minority instances (He et al., 2008; Bellinger et al., 2019, 2021; Hassanat et al., 2022). A further limitation of the majority - if not the entirety - of the state-of-the-art approaches is that they (implicitly) exploit the number of instances belonging to a specific class to characterize differences and similarities among instances, even though features should be the ones capturing these differences/similarities.

To overcome the weaknesses of state-of-the-art approaches, we propose FROID a pre-processing framework for *Features Reduction and OutlIer Detection* that allows solving the imbalanced learning problem through unsupervised representation learning. FROID handles imbalanced learning by facing the problem from a different perspective. Indeed, instead of augmenting the instances of the minority classes or reducing the instances of the majority classes, FROID analyzes the relationships among the records in the dataset through unsupervised approaches. The goal of FROID is to design attributes creating an unsupervised data representation that enhance the differences between records belonging to minority and majority classes such that a ML model can achieve outstanding performance regardless of the class imbalance.

In particular, FROID exploits two families of methods to augment the expressiveness of the records in a dataset. The first family of methods is the one of *Outlier Detection* (OD) approaches (Chandola et al., 2009; Hodge & Austin, 2004). An unsupervised OD approach is meant to identify outliers, i.e., instances which deviate significantly from the majority of the data and do not conform to a notion of normal behavior (Chandola et al., 2009). Our intuition is that records belonging to minority classes should be considered as outliers with respect to records belonging to majority classes. Therefore, the usage of unsupervised OD methods can create attributes capturing the level of *outlierness* of a record with respect to other records concerning a certain OD criterion. A similar intuition to solve the task of supervised OD was proposed in Zhao and Hryniewicki (2018). However, in Zhao and Hryniewicki (2018) unsupervised OD methods are used to boost a supervised OD problem. Thus, it is known form the problem definition that there are outliers in the data. On the other hand, in our case, we are just making a supposition that instances belonging to minority class can be recognized as outliers by unsupervised OD approaches. The second family of methods comprehends *Features Reduction* (FR), also known as features projection, features extraction, or dimensionality reduction. FR methods transform the data from a high-dimensional space to a space of fewer dimensions. The data transformation may be linear, as in Principal Component Analysis (Pearson, 1901), but many nonlinear dimensionality reduction techniques also exist (Cox & Cox, 2008; Van der Maaten & Hinton, 2008; Tenenbaum et al., 2000). Similar to the reasons that brought us to rely on OD approaches, our idea is that unsupervised FR techniques might unveil a data representation that better separates among instances belonging to different classes. Indeed, rare instances should acquire a reduced data representation substantially different from that of the instances belonging to the regular class that, on the other hand, should fall in denser areas of the reduced representation. In the literature, there is a limited set of methods relying on FR to address imbalanced learning problems (Naseriparsa & Kashani, 2014; Gopi et al., 2016). However, these approaches only rely on a unique FR method and combine it with resampling techniques. On the other hand, we adopt a large array of FR approaches, and we do not augment the number of records in the dataset. Also, FROID subsequently combines the outcomes of OD and FR approaches through several workflows to create more and more expressive features to separate records of different classes for the classification task.

We experimented with FROID on 64 benchmarking datasets and 2 case studies by training 5 different ML models after pre-processing the data with FROID. First, we observed which type of classifier benefits more from the pre-processed data returned by FROID. Second, we performed an ablation study of FROID showing which is the impact of every set of features extracted by the various OD and FR approaches. Third, we compared FROID with some state-of-the-art techniques to deal with imbalanced learning. The results show that *(i)* on average LigthGBM (Ke et al., 2017) is the best classifier exploiting the unsupervised representation returned by FROID, *(ii)* the more features are extracted through FROID, the higher are the performance of the classifier, and *(iii)* FROID outperform all the state-of-the-art approaches at the cost of a not negligible running time required to extract all the features. Finally, we highlight that, besides imbalanced learning, FROID also succeeds in the supervised outlier detection task.

The rest of the paper is organized as follows. Section 2 reviews related works of imbalanced learning and of supervised outlier detection. In Sect. 3 we illustrate our proposal to solve imbalanced learning through outlier detection and features projection approaches. Section 4 reports the experimental results on benchmarking datasets as well as on two case studies. Finally, Sect. 5 summarizes our contributions and discusses open research directions.

2 Related works

A large array of approaches have been proposed to face imbalanced learning. In the last twenty years, several surveys and literature reviews have categorized and discussed peculiarities and characteristics of the various approaches (Japkowicz & Stephen, 2002; Chawla et al., 2004; Su & Tsai, 2011; He & Garcia, 2009; Chawla, 2010; Branco et al., 2016). Recently, most of these approaches have been implemented and are freely available in Python open-source libraries like `imblearn` (Lemaitre et al., 2017). The two principal strategies to recover from the challenges raised by imbalanced learning contexts consist in modifying the data used to train the classifier, or altering the classification algorithm itself to account for misclassification costs of the different classes during the learning process. Random undersampling and oversampling (Kubat & Matwin, 1997) are the two most classic approaches to handling imbalance. It is well known that they suffer from the risk of discarding informative instances and overfitting the minority instances, respectively. Refinement of undersampling techniques like the Condensed Nearest Neighbors (CNN) (Hart, 1968) or the Edited Nearest Neighbors (ENN) (Wilson, 1972) brought slight improvements to such issues but not effective solutions. Nowadays, the Synthetic Minority Oversampling Technique (SMOTE) (Chawla et al., 2002) is probably the most widely used, exploited, and extended oversampling approach. For instance, ADASYN (He et al., 2008) is very similar to SMOTE, but it generates a different number of samples depending on an estimate of the local distribution of the class to be oversampled. SVMSMOTE (Nguyen et al., 2011) exploits an SVM algorithm to detect the samples to use for generating the synthetic instances for oversampling the minority class.

Besides improving the procedure of these first resampling approaches, one of the most pursued research directions consists in combining them with other Data Mining or Machine Learning approaches such as clustering algorithms or simple classification models. For instance, the ClustFirstClass undersampling approach (Sobhani et al., 2014) tries to overcome the problem of discarding informative instances by first running the k-Means clustering (Tan, 2005) on the majority class and then at least one instance is maintained from each cluster. In Sundarkumar and Ravi (2015), the majority class outliers are removed with Reverse k-Nearest Neighborhood (RkNN) (Achtert et al., 2006). Then, the selection of support vectors using the One-Class Support Vector Machine (OCSVM) (Schölkopf et al., 1999) is used to undersample the majority class. In Sanguanmak and Hanskunatai (2016) is presented the DBSM approach for simultaneous undersampling and oversampling. The oversampling is performed with SMOTE, while the undersampling is realized by selecting half of the data which are present in the clusters returned by the DBSCAN method (Ester et al., 1996). A similar solution is presented in Branco et al. (2018) with the idea of biasing the strategies to reinforce some regions of the datasets instead of sampling uniformly. Such biases are applied through random undersampling and SMOTE. In Douzas et al. (2018) is presented k-SMOTE a refinement of SMOTE that exploits k-Means to avoid the generation of noisy synthetic minority instances which are erroneously close to a dense area of records belonging to the majority class. SMOTEFUNA is a further refinement of SMOTE presented in Tarawneh et al. (2020). SMOTEFUNA generates synthetic records between a randomly selected instance and its furthest neighbor of the minority class which have not a nearest neighbor of the majority class. Instances of the majority class are considered also in Koziarski and Wozniak (2017); Koziarski et al. (2021). Indeed, in Koziarski and Wozniak (2017) is proposed the CCR algorithm that Combines Cleaning of the decision border around minority objects with guided synthetic Resampling to re-balance

the dataset. In Koziarski et al. (2019) is proposed the Radial-Based Oversampling method (RBO) that discovers regions in which the synthetic objects from minority class should be generated with radial basis functions. An extension of CCR and RBO is presented in Koziarski et al. (2021): Radial-Based CCR exploits the class potential to locate sub-regions of the data-space for synthetic oversampling and adopts radial basis functions. The ClUstered REsampling (CURE) method (Bellinger et al., 2019) uses hierarchical clustering and a newly defined distance measure to guide the resampling procedure. Such clusters take into account the structure of the data. This aspect enables CURE to avoid the generation of synthetic instances in "wrong" regions, and allows the undersample of non-borderline regions of the majority class. In Bellinger et al. (2021) is proposed ReMix, a pre-processing approach that leverages batch resampling and instance mixing to enable the induction of robust deep models from imbalanced and long-tailed datasets by expanding the minority class to reduce predictive bias. The objective of ReMix is not only to improve the predictive performance but also to increase model calibration. In Sharma et al. (2018) SWIM (Sampling WIth the Majority) is presented as a method for synthetic oversampling that exploits the information inherent in the majority class to synthesize minority class records. In a certain sense, we use an idea similar to SWIM because the objective of FROID is to enhance discriminative aspects among minority an majority records by looking to both of them and not only to minority records. Finally, the most widely studied case study for imbalanced learning is *fraud detection* (Padmaja et al., 2007; Makki et al., 2019; Tran et al., 2021, 2021; Esenogho et al., 2022). In all these works, besides experimenting with existing techniques, further refinements and extensions are proposed but all focusing on data resampling.

From the analysis of these approaches, we noticed the following aspects. First, they all focus only on recovering issues of previous versions. Second, the usage of additional mining or learning techniques brings some benefits but also problems related to the hyper-parameter tuning. Third, and most importantly, all these methods account only for the "number of instances" dimension of a dataset and leave unaltered the features used to represent the records in the dataset. In this paper, we define an approach to modify the data used to train the classifier, therefore falling in this aforementioned category of papers. However, we focus on the features used instead of the instances that should be present in the training set, and we avoid any hyper-parameter tuning.

As stated at the beginning of this section, the second and less followed line of research is relative to making crucial changes in the classification algorithm in order to account for imbalanced class scenarios. In Akbani et al. (2004) is proposed an upgrade of SVM based on a variant of SMOTE and combined with the error costs presented in Veropoulos et al. (1999) which penalizes more misclassifications w.r.t. minority instances than misclassifications to majority instances. The DataBoost-IM method (Guo & Viktor, 2004) identifies instances which are *hard* to classify through boosting approaches. Then it trains an ensemble-based boosting algorithm by generating synthetic instances with biased information toward the *hard* instances on which the next classifier in the boosting procedures needs to focus. In Wang et al. (2014) is presented an instance-weighted variant of the SVM with both 1-norm and 2-norm formats to deal with imbalanced learning. Also none of these approaches account for the features describing the records.

Conversely, the following approaches also account for this delicate aspect. In Naseriparsa and Kashani (2014), the Principal Component Analysis (PCA) (Tan, 2005) feature projection approach is combined with SMOTE in a case study on a single dataset. The procedure first applies PCA to the dataset, then SMOTE for each of the minority features before applying the classification algorithm. In Gopi et al. (2016) is defined a Support

Vector Machine-Recursive Feature Elimination (SVM-RFE) wrapper for feature selection. The Automated Imbalanced Classification (ATOMIC) method presented in Moniz and Cerqueira (2021) is an Automated Machine Learning (AutoML (He et al., 2021)) approach for imbalanced classification that extends the features describing the data with additional statistical features. In Ksieniewicz (2019), an ensemble of classifiers is trained on datasets obtained as random subspace and augmented through SMOTE. In Korycki and Krawczyk (2021), a similar methodology is applied to find the most discriminative low-dimensional representation instead of a random one. Since all these approaches do not increase the number of features and yet improve the performance of "traditional" approaches for imbalanced learning, our idea is to follow this intuition but augment the features used to represent the data in order to maximize the difference between instances belonging to different classes.

Another research field related to our proposal is outlier detection (Hodge & Austin, 2004; Chandola et al., 2009). Indeed, our intuition is that instances belonging to the minority class can be considered to some extent as outliers w.r.t. instances belonging to the majority class. In supervised outlier detection, a predictive model is trained on a dataset that has labeled instances for normal as well as anomaly classes. Thus, our intuition is in line with the XGBOD method presented in Zhao and Hryniewicki (2018). Indeed, XGBOD uses multiple unsupervised outlier detection algorithms to extract an alternative representation of the instances that augment the predictive capabilities of XGBOOST (Chen & Guestrin, 2016) to solve supervised outlier detection. The same intuition is followed by the geodesic-based outlier detection method which considers Global Disconnectivity score and Local real Degree (GDLD) as measures of outlierness presented in Shi et al. (2020). Indeed, GDLD is evaluated in the imbalanced learning setting considering records belonging to the minority class as outliers. In Shimauchi (2021) is presented a semi-supervised outlier detection algorithm that extends XGBOD through the augmentation of the representations with a Generative Adversarial Network (GAN) (Goodfellow et al., 2014). In Fernández et al. (2022) is proposed a framework for supervised outlier detection that is formed by a pipeline with an unsupervised outlier detection followed by a supervised predictive model that is used to tune the hyper-parameters of the unsupervised outlier detection algorithm. Finally, it is worth mentioning the approach illustrated in Loureiro et al. (2004) that applies in a case study, an unsupervised outlier detection method based on hierarchical clustering. From our perspective, the interesting aspect of Loureiro et al. (2004) is that it employs an unsupervised clustering-based strategy similarly to works previously discussed to solve imbalanced learning. Thus, it supports our intuition that solving imbalanced learning also through approaches used for outlier detection is a viable path. ODBOT, an alternative to XGBOD, i.e., an outlier detection-based oversampling technique for imbalanced datasets learning is presented in Ibrahim (2021). ODBOT handles multi-class imbalanc by finding clusters within the minority class(es), and then, generating synthetic samples by consideration the outliers detected in these clusters. Finally, we underline how in Hassanat et al. (2022) is shown that oversampling methods are not reliable. Indeed, the authors of Hassanat et al. (2022) reports an experimentation on more than 70 oversampling methods that reveal that the oversampling methods studied generate minority samples that are most likely to be majority. Hence, oversampling methodologies are quite likely to be unreliable in imbalanced learning settings and should be avoided in real-world applications.

While works like (Shi et al., 2020; Ibrahim, 2021; Ksieniewicz, 2019; Korycki & Krawczyk, 2021) already introduced the idea of boosting imbalanced learning methods through outlier detection or feature reduction, we stress the fact that, to the best of our knowledge, our proposal is the first in which they are used simultaneously and into subsequent iterations. Furthermore, our proposal departs from Shi et al. (2020); Ibrahim (2021);

Ksieniewicz (2019); Korycki and Krawczyk (2021) because they always augment the number of instances, while FROID leaves their number unaltered and plays only with the different data representations in order to leverage the discriminative power of ML models.

3 Methodology

In this paper we present FROID, a Features Reduction and OutlIer Detection pre-processing framework for solving imbalanced learning through unsupervised representation learning. FROID is a pre-processing framework that takes as input a dataset X and returns a transformed version of it to be used as training for a ML classification algorithm. The main idea of FROID is to represent the instances in X through alternative representations aimed at fostering the differences among instances belonging to different classes. Thus, FROID relies on *Outlier Detection* (OD) and on *Features Reduction* (FR) approaches.

In Fig. 1 we illustrate an example that visualizes our intuition of representing the imbalanced input data through unsupervised techniques of OD and FR. The first plot (top left) depicts a synthetically generated imbalanced dataset with two dimensions and two classes with frequencies.95 and.05, respectively. In the second plot (top right), is illustrated the decision boundary learned by a Decision Tree classifier (Tan, 2005) trained on the imbalanced dataset. We immediately notice how most of the instances belonging to the minority class and located in the range $X_0 \in [-1, 2]$ and $X_1 \in [-1, 1]$, highlighted by the yellow rectangle, are wrongly classified as majority instances by the Decision Tree. The third plot (bottom left) represents the synthetic dataset using as features the Local Outlier Factor (Breunig et al., 2000) score (LOF), an unsupervised OD approach, and the first Principal Component returned by the Principal Component Analysis (Pearson, 1901) (PCA) approach that is a FR method. We notice how the instances of the minority class are now displaced along two parallel horizontal directions. The fourth plot (bottom right) shows the decision boundary of a Decision Tree trained on this novel representation with the same parameter setting as the previous one. The rare instances inside the yellow rectangle in the second plot are represented with yellow squares in the fourth plot and are located approximately in the ranges $LOF \in [-.8, -.4], PCA \in [1.0, 1.5]$ and $LOF \in [-.8, -.46], PCA \in [-2.5, -.5]$. Among these rare instances, we notice that only three are not covered by decision rules labelling instances as minority class, i.e., green decision boundaries areas, and are therefore misclassified as majority class instances. Hence, by representing a two-dimensional dataset through an OD score and an FR dimension, we have improved the performance of an ML model, passing from an F1 measure of.60 to an F1 measure of.64. This simple example simultaneously explains and proves the intuition behind the proposed idea.

3.1 Imbalanced learning pre-processing framework

In this section, we define the Features Reduction and OutlIer Detection pre-processing framework FROID to solve imbalanced learning through unsupervised representation learning.

Problem setting The FROID framework is depicted in Fig. 2 and highlighted with the dashed box. FROID works as detailed in the following. Let $X \in \mathcal{R}^{n \times m}$ denote the original input dataset as a set of n instances with m records. Each record $x_i \in X$ has attached a label $y_i \in [0, \ldots, l]$ indicating the class of the record where l is the number of the classes.

However, since FROID is an unsupervised representation learning framework, the class y is not used. On the other hand, FROID makes usage of:

- A set of u Outlier Detection (OD) functions $\Theta = \{\theta_1, \ldots, \theta_u\}$
- A set of v Feature Reduction (FR) functions $P = \{\rho_1, \ldots, \rho_v\}$
- A features selection function ζ

Outlier detection transformation We define an OD function θ_j as a mapping function where the output can be a real-valued vector $\theta_j(X) \in \mathcal{R}^{n\times 1}$ that describes the degree of outlierness of each instance $x_i \in X$. Outliers are instances that deviate significantly from the majority of the data and do not conform to a notion of normal behavior (Chandola et al., 2009). We include in this representation the cases in which the OD function θ_j returns a binary-valued vector $\theta_j(X) \in \{0, 1\}^{n\times 1}$ that indicates if the i^{th} instance is an outlier or not. We indicate with $X_o \in \mathcal{R}^{n\times u}$ the result of the application of the u OD functions on X, i.e., $X_o = [\theta_1(X), \ldots, \theta_u(X)]$. Details of the OD approach implementing the OD functions adopted are provided in the next section.

Features reduction transformation We define an FR function ρ_j as a mapping function where the output $\rho_j(X) \in \mathcal{R}^{n\times p}$ describes the representation of each instance $x_i \in X$ into a p-dimensional space. FR methods transform the data from a high-dimensional to a low-dimensional space. The lower dimensionality aims to capture salient aspects of the higher dimensionality. We indicate with $X_r \in \mathcal{R}^{n\times(vp)}$ the result of the application of the v FR functions on X, i.e., $X_r = [\rho_1(X), \ldots, \rho_v(X)]$. Details of the FR methods implementing the functions adopted are in the next section.

Features selection transformation We define a features selection function ζ as a mapping function $\zeta(X) : \mathcal{R}^{n\times c} \rightarrow \mathcal{R}^{n\times k}$ that reduces the dimensionality of a given input $X \in \mathcal{R}^{n\times c}$ to an output $X' \in \mathcal{R}^{n\times k}$ where $k < c$, i.e., ζ remove some of the c columns from the input X, i.e., $X' = \zeta(X)$.

Workflow description Given the input dataset X, the OD functions Θ, the FR function P, and the features selection function ζ, we can recognize three phases in the preprocessing performed by FROID. We define an OD function θ_j as a mapping function where the output can be a real-valued vector $\theta_j(X) \in \mathcal{R}^{n\times 1}$ that describes the degree of outlierness of each instance $x_i \in X$. Outliers are instances that deviate significantly from the majority of the data and do not conform to a notion of normal behavior (Chandola et al., 2009). We include in this representation the cases in which the OD function θ_j returns a binary-valued vector $\theta_j(X) \in \{0, 1\}^{n\times 1}$ that indicates if the i^{th} instance is an outlier or not. We indicate with $X_o \in \mathcal{R}^{n\times u}$ the result of the application of the u OD functions on X, i.e., $X_o = [\theta_1(X), \ldots, \theta_u(X)]$. Details of the OD approach implementing the OD functions adopted are provided in the next section.

In the *first phase*, the OD and FR functions Θ and P are applied to X obtaining X_o and X_r, respectively. In the *second phase*, the OD and FR functions Θ and P are applied subsequently on X_o and X_r obtaining the following representations:

- $X_{oo} = [\theta_1(X_o), \ldots, \theta_u(X_o)]$
- $X_{ro} = [\rho_1(X_o), \ldots, \rho_v(X_o)]$
- $X_{rr} = [\rho_1(X_r), \ldots, \rho_v(X_r)]$
- $X_{or} = [\theta_1(X_r), \ldots, \theta_u(X_r)]$

In the *third phase*, all the data representation obtained together with the input dataset X are concatenated and passed again to the features selection operator resulting in

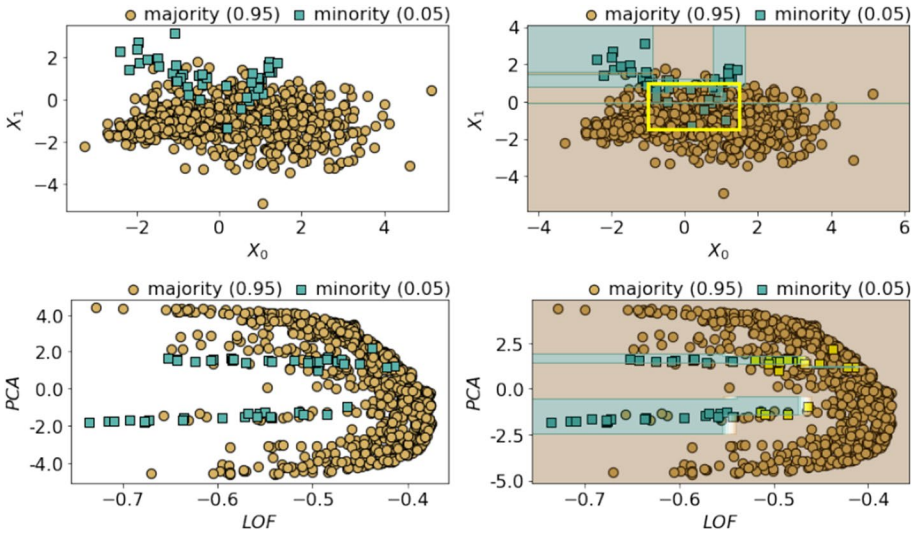

Fig. 1 Utility of representing a dataset through OD and FR scores. Top left: synthetically generated imbalanced dataset with two dimensions and with two classes. Top right: decision boundary learned by a Decision Tree trained on the imbalanced dataset. Bottom left: synthetic dataset represented through LOF and PCA. Fourth plot: decision boundary of a Decision Tree trained on the unsupervised alternative representation (Color figure online)

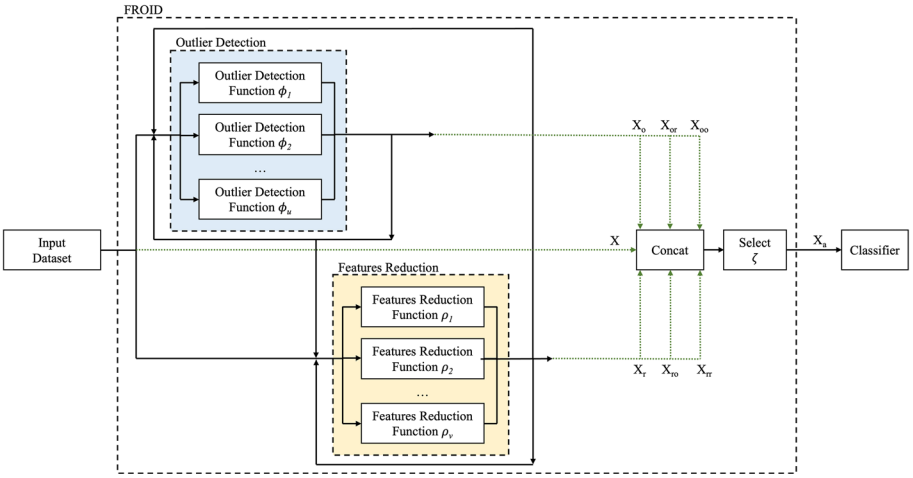

Fig. 2 Illustration of FROID framework for features extraction. The input dataset X is passed through a set of unsupervised outlier detection functions θ_i (in blue) and through a set of unsupervised features reduction functions ρ_i (in yellow) originating the datasets X_o and X_r, respectively. Such datasets are passed again through the functions θ_i and ρ_i originating the datasets $X_{oo}, X_{or}, X_{rr}, X_{ro}$, respectively. Finally, all the datasets are combined and the best features are selected from the combination with ζ (Color figure online)

$X_a = \zeta([X, X_o, X_r, X_{oo}, X_{ro}, X_{rr}, X_{or}])$. We highlight how FROID never augments the number of records in the dataset along with the various phases and data transformation, i.e., $|X| = |X_a|$, differently from all the other state-of-the-art approaches for the class imbalance

problem. On the other hand, the result of the unsupervised pre-processing $X_a \in \mathcal{R}^{n \times m'}$ is a data representation using several features m' that is unknown a priori because it strictly depends on the data transformation functions Θ, P, ζ employed. In the end, any ML model can be trained on X_a, y. We underline that when selecting a heterogeneous sets of OD methods Θ and FR methods P, we can guarantee that every record in X will be represented w.r.t. different data-driven criteria. In addition, the risk of creating correlated features is effectively minimized, if not entirely eliminated, through *(i)* the utilization of a feature selection function, denoted as ζ, and *(ii)* the adoption of tree-based approaches as final classifiers. Thus, through this approach, we are able to guarantee that the representation of each record in X_a is diverse and distinct, enabling ML models to capture various aspects and characteristics of the data to amplify the separation between records of majority and minority classes.

In the rest of this section, we illustrate the OD and FR methods we considered to implement the functions of the FROID pre-processing framework.

3.2 Outlier detection methods

We consider a large set of OD methods to implement the OD functions Θ. In particular, we rely on u=14 OD methods based on different ideas and strategies to assign an outlier score/label to a given instance. In the following, we briefly describe the selected OD methods, which are highlighted in bold.

A large family of OD methods relies on the notion of *locality*, i.e., the outlier score is assigned by comparing a record with its neighbors with respect to a distance function and neighborhood size k. **kNN**, k-Nearest Neighbor (Tan, 2005) is a supervised ML algorithm frequently used for classification problems. However, it can also be used as an OD method by returning as outlier score of a record the largest distance from the instances in its kNN. **LOF**, Local Outlier Factor (Breunig et al., 2000) assigns an outlier score by comparing the local density of a record with the local densities of its kNN. If a record lies in an area with a density substantially lower than its neighbors, then it is considered an outlier. **LoOP**, Local Outlier Probability (Kriegel et al., 2009) is a local density-based OD method that extends LOF by measuring the local deviation of the density of a given instance with respect to its neighbors as LOF scores. It can work directly on the input data or on the result of a clustering algorithm by relating the outlier score calculus to the distances of the clusters' centroids. **COF**, Connectivity-Based Outlier Factor (Pokrajac et al., 2008), overcome some limitations of LOF by calculating the outlier score of a record as its degree of *connectivity*. COF differs from LOF as it uses the chaining distance to calculate the kNN. The chaining distances are the minimum of the total sum of the distances linking all neighbors. The connectivity is then calculated as the ratio between the average chaining distance of the record and the mean average chaining distance of the records in the kNN. An additional possibility we explored for implementing the OD functions is to employ clustering approaches that highlight instances not belonging to any cluster as outliers (Khan et al., 2014). **CBLOF**, Cluster-Based Local Outlier Factor (He et al., 2003) takes as input both the dataset and a clustering algorithm and labels each cluster as "small" or "large" with respect to two α and β parameters. The outlier score of a certain instance is then calculated w.r.t. the size of the cluster the point belongs to and the distance to the nearest "large" cluster.

Another family of OD approaches exploits *global* statistical tests and global models to discover anomalous behaviors. The **Elliptical Envelope** (Rousseeuw & van Driessen, 1999) algorithm (EllEnv) creates a global elliptical area that surrounds input data. Values

that fall inside the envelope are considered normal data, and anything outside is considered an outlier. **OCSVM**, One-Class SVM (Schölkopf et al., 1999) is a variation of Support Vector Machines (SVM) (Tan, 2005) that can be used in an unsupervised setting for OD. The idea of OCSVM is to find a function that is positive for regions with a high density of points, and negative for small densities, considering the records that fall into negative regions of the hyperplane as outliers. **MCD**, Minimum Covariance Determinant (Hubert & Debruyne, 2010) is commonly applied on Gaussian-distributed data. MCD fits a minimum covariance determinant model (Hubert et al., 2018) and computes the outlier score through the Mahalanobis distance calculation. **HBOS**, Histogram-Based Outlier Detection (Goldstein & Dengel, 2012) assumes feature independence and calculates the outlier scores by building histograms. **COPOD**, Copula-Based Outlier Detection method (Li et al., 2020), instead, creates an empirical copula and uses it to predict each record's tail probabilities to determine its outlier score.

An efficient and effective OD approach consists of using an ensemble of "weak" OD methods. The Feature Bagging (**FeaBag**) (Lazarevic & Kumar, 2005) exploits a set of OD methods, each of them applied on a random set of features selected from the original feature space. Each OD method identifies different outliers and assigns to all instances outlier scores that correspond to their probability of being outliers. The combination of such scores is returned as the final output. **Isolation Forest** (Liu et al., 2008) (IsoFor) is one of the most famous OD approaches. IsoFor isolates instances by randomly selecting a feature and then randomly selecting a split value in the range of the feature. This process is represented through a tree where the number of splittings required to isolate a record equals the path length from the root node to the leaf. Hence, an instance is considered an outlier when a forest collectively produces shorter path lengths for that instance. An extension of HBOS is **LODA**, Lightweight On-line Detector of Anomalies (Pevný, 2016). LODA approximates the joint probability using a collection of one-dimensional histograms, where every one-dimensional histogram is efficiently constructed on an input space projected onto a randomly generated vector. Even though one-dimensional histograms are weak OD methods, their collection yields a strong OD approach. **SUOD**, Scalable Unsupervised Outlier Detection (Zhao et al., 2020) is another OD ensemble method. Given in input a dataset and a set of unsupervised OD methods, SUOD randomly projects the original input onto lower-dimensional spaces and speeds up the training through a balanced parallel scheduling to assign averaged outliers scores.

3.3 Features reduction methods

We consider a wide set of FR methods to implement the FR functions P. We rely on $v = 8$ FR methods described in the following and highlighted in bold.

Most of the approaches in the literature are based on the idea of finding novel directions along which the original data should be projected. The various techniques differ on how these directions are built or derived. **PCA**, Principal Component Analysis (Pearson, 1901; Hasan & Abdulazeez, 2021) is the process of computing the principal components and using them to perform a feature projection of the data along with these principal directions. Indeed, a principal component is the direction of the line that best fits the data while being orthogonal to the previous component. Principal components, therefore, are the derived variables formed as a linear combination of the original variables that explains the most variance. **MDS**, MultiDimensional Scaling (Cox & Cox, 2008) is a process that translates the records of a given high-dimensional dataset into a low-dimensional representation with

respect to the pairwise distances observed in the original space: instances which are close in the original space should be close also in the reduced space, and vice-versa. **IsoMap**, Isometric Features Mapping (Tenenbaum et al., 2000) is a nonlinear dimensionality reduction method. IsoMap estimates the intrinsic geometry of a data manifold by estimating the geodesic distance between all pairs of instances on a weighted graph built with respect to the nearest neighbor identified through a fixed radius. The top eigenvectors of the geodesic distance matrix represent the coordinates in the new reduced space. **LLE**, Locally Linear Embedding (Roweis & Saul, 2000) is similar to IsoMap, but instead of using the geodesic distance, it uses a distance based on the ability to reconstruct a record with respect to its neighbors. A well-known issue of LLE is the regularization problem. A way to address it is to use different methods for **LLE**, for instance the Modified LLE (Zhang & Wang, 2006), or the **HLLE**, Hessian eigenmapping LLE (Donoho & Grimes, 2003). **SpectEmb**, Spectral Embedding (Bengio et al., 2006) is exploited for non-linear dimensionality reduction using a spectral decomposition of a the graph modeling the dataset. Although SE is similar to IsoMap and LLE, it differs in how the weights are calculated, and it adopts the eigenvectors returned from a Laplacian Matrix as reduced dimensionality. Finally, we employed **t-SNE**, t-distributed Stochastic Neighbor Embedding (Van der Maaten & Hinton, 2008) is a form of MDS that, besides preserving the distances, also aims at preserving the neighborhoods of the instances by modeling the distances as probability distributions belonging to a certain neighborhood.

4 Experiments

We report here the experiments carried out to validate FROID. First, we illustrate the experimental setting with the datasets used, the classifiers adopted, the implementations and parameters employed, the competitors analyzed, and the evaluation measures tested. Second, we show which is the best ML classifier among the various datasets and the improvement of FROID w.r.t. training the models on the original data. Third, we report an ablation study of the unsupervised features adopted by FROID. Fourth, we compare FROID with state-of-the-art solutions. Fifth, we prove that the pre-processing of FROID is beneficial also for supervised outlier detection. Finally, we discuss which are the most important features adopted by FROID in two real case studies.

4.1 Experimental setting

In this section, we illustrate the experimental setting with the datasets used, the classifiers adopted, the implementations and parameters employed, the competitors analyzed, and the evaluation measures tested.

4.1.1 Datasets and machine learning classifiers

We ran experiments on a selection of 64 binary classification datasets widely referenced and used for imbalanced learning experiments publicly available from the UCI, Kaggle,

ODDS, KEEL and `imblearn` repositories and a fraud detection challenge.[1] For each dataset, the following pre-processing is applied. First, we remove records with null values without replacement not to compromise the originality of the data. Next, we eliminate columns with poor explanatory potential, such as IDs, names, etc. Categorical columns are encoded through one-hot encoding to preserve the semantic meaning of the variables for usage with OD and FR methods based on distances or vectors and also for the correctness w.r.t. the ML models.[2] Datasets description after this pre-processing, as well as some data complexity measures (Sotoca et al., 2005; Cano, 2013), are available in Table 11 in the Appendix.[3] We summarize the information contained in Table 11 by running K-Means with $k = 4$ to group the different types of datasets analyzed and provide a brief description of the datasets. Indeed, in Table 1, we report a summary of the datasets through the centroids of the four clusters. We observe that the majority of the datasets is "small-sized" and with the lowest *FDR* (cluster *A*). In contrast, the other larger datasets are further separated either w.r.t. the dimensionality or w.r.t. *FBP*.

Before training the ML models or running the imbalanced learning pre-processing solutions, we applied to the datasets the *Robust Scaler* that normalizes the features using statistics that are robust to outliers.[4] The Robust Scaler removes the median and scales the data with respect to the Interquartile Range (IQR), i.e., the difference between the 3^{rd} quartile (75^{th} quantile) and the 1^{st} quartile (25^{th} quantile). This choice is tied with better discrimination among instances belonging to minority or majority classes. Indeed, the Robust Scaler normalizes values, and those far away from the median value and outside the IQR will get values markedly greater/smaller than zero. If datasets still have to be partitioned into training and test, we split them using a stratified hold-out partitioning based on the target class, with 70% of the data used for the training and 30% for the test. Otherwise, we keep the original train-test partitioning. To guarantee a statistically valid evaluation, as proposed in Rajkomar et al. (2018), we bootstrapped each test set 100 times, and we report in the manuscript the mean values obtained by the various classifiers over these runs.

As ML classifiers, due to the proven empirical superiority of ensemble models (Breiman, 2001; Shwartz-Ziv & Armon, 2022), we decided to experiment with Decision Tree (Breiman et al., 1984) (DT), Random Forest (Breiman, 2001) (RF), XGBoost (Chen & Guestrin, 2016), LightGBM (Ke et al., 2017), and CatBoost (Prokhorenkova et al., 2018) as implemented by the `sklearn`, `xgboost`, `catboost`, and `lightgbm` Python libraries.[5] If not differently specified, we adopted the default parameter setting proposed by the various libraries to assess to which extent different pre-processing techniques are more effective for solving imbalanced learning with the same hyperparameter values.

[1] https://archive.ics.uci.edu/, https://www.kaggle.com/datasets, http://odds.cs.stonybrook.edu/, https://generali.datachallenge.it/, https://sci2s.ugr.es/keel/datasets.php, https://imbalanced-learn.org/stable/datasets/index.htm.

[2] We highlight that, even though the input domain of OD and FR methods is typically continuous and not binary as the one returned by one-hot encoding, we are satisfied as far as it positively contributes to helping in separating instances belonging to different classes in the imbalanced learning scenario.

[3] The `ecoli` `glass`, and `satal` datasets are used both for imbalanced learning and for supervised outlier detection.

[4] https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.RobustScaler.html.

[5] https://scikit-learn.org/, https://catboost.ai/, https://xgboost.readthedocs.io/.

4.1.2 Experimental details

We *implemented* FROID in Python[6] by relying on the following libraries. The core of the algorithm is realized following the `scikit-learn` style and adopting the notion of *pipeline* such that every OD or FR method can be subsequently enabled or disabled. For the OD methods we relied on the implementations of the Python libraries `sklearn`, `pyod` (Zhao et al., 2019), and `PyNomaly`,[7] while for FR methods on the implementations offered by the Python library `sklearn`. The mapping between OD and FR methods and the selected implementations with the parameters varied is reported in Table 2. For instance, the LoOP method is implemented with the `pynomaly` library and used with the parameter number of neighbors $k \in [1, 5, 10, 20]$. Among the selected implementations of OD functions Θ, given an instance x_i, all of them can be used both for returning a binary value indicating if x_i is an outlier or not and for returning the degree of outlierness of x_i. Hence, the number of features extracted by FROID through OD methods is theoretically $2u$ where $u = |\Theta|$. However, in practice, such a number is higher than $2u$ and depends on the parameter combinations used for each OD method.[8] On the other hand, with all the FR methods we projected the input data into $p = 2$ dimensions.[9] Finally, if not differently specified, the features selection function ζ is implemented with the `sklearn` library[10] that trains a LightGBM model on the dataset and selects only the features having importance higher than the average. As an alternative, we also implement ζ with the variance threshold function[11] that removes from the input dataset all low-variance features below a certain threshold.[12]

We underline that, for every dataset, a different number of features might be generated by FROID because some parameters configurations for OD and FR methods might be invalid depending on the dataset characteristics. Also, some of the classifiers implementations are not able to handle missing and/or too large values. In this case, FROID drops all the generated featured that meet one of these conditions. On average, we observed that FROID generates about 508 features per dataset.

[6] The code of FROID is available here https://github.com/andrepugni/FROID.

[7] https://pyod.readthedocs.io/https://github.com/vc1492a/PyNomaly.

[8] For OD methods we varied the following parameters: for LoOP and COF we used $k \in \{1, 5, 10, 20\}$ - due to computational reasons - while for kNN and LOF we used $k \in \{1, 2, 3, 4, 5, 10, 15, 20, 30, 40, 50, 60, 70, 80, 90, 100, 150, 200, 250\}$; for kNN we also considered different method for distance - $method \in \{mean, median, largest\}$ - and for IsoFor we employed different number of estimators, i.e. $n \in \{0, 20, 50, 70, 100, 150, 200, 250\}$; the *contamination* parameter α controls the percentage of outliers that we expect in the dataset and affects the binary value returned by OD functions $\alpha \in \{.001, .01, .1, .2, .5\}$; the *extent* parameter controls the probabilistic distance of a instance to a context set (Kriegel et al., 2009) $extent \in \{1, 2, 3\}$; the ν parameter controls the upper bound on the fraction of training errors and a lower bound of the fraction of support vectors (Schölkopf et al., 1999) $\nu \in \{.01, .1, .2, .3, .4, .5, .6, .7, .8, .9, .99\}$.

[9] For FR methods we varied the following parameters: the *kernel* parameter controls the kernel type (Pearson, 1901) $kernel \in \{rbf, cosine, sigmoid, poly\}$. For the MDS we used $max\_iter = 100$, for LLE we also considered $method = hessian$ and $method = modified$. We used default values for the parameters not specified in the above lists and table.

[10] https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectFromModel.html.

[11] https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.VarianceThreshold.html.

[12] Some of the classifiers implementations can not handle missing and too large values. In this case, we dropped all the generated featured meeting one of these conditions.

Table 1 Dataset description aggregated through K-Means clustering: n_{train} instances training set; n_{test} instances test set; m number of features; m_{num} number numerical features; p^{+}_{train} positive rate training; p^{+}_{test} positive rate test set; *FDR* Maximum Fisher's Discriminant Ratio, *FBP* Fraction of Borderline Points, *ECP* Entropy of Class Proportions and *IR* Imbalance Ratio

Cluster	n_{train}	n_{test}	m	m_{num}	p^{+}_{train}	p^{+}_{test}	*FDR*	*FBP*	*ECP*	*IR*	Size
A	1439.75	617.73	23.37	18.92	.062	.062	.759	.062	.731	.894	56
B	8782.00	3764.33	29.66	29.66	.051	.053	.919	.114	.710	.890	3
C	5457.00	234.00	617.00	617.00	.076	.076	.953	.064	.608	.834	1
D	19323.50	9773.75	11.75	11.75	.042	.042	.952	.048	.754	.909	4

Table 2 Mapping between OD and FR methods used by FROID and the selected implementations with the parameters varied: `sklrn` means `sklearn`, `pynomaly` means `pynml`

OD			OD			FR		
Name	Library	Params	Name	Library	Params	Name	Library	Params
kNN	`pyod`	$k, \alpha, method$	SUOD	`pyod`	α	PCA	`sklrn`	$p = 2$
LOF	`sklrn`	k	LODA	`pyod`	α	IsoMap	`sklrn`	$p = 2$
LoOP	`pynml`	$k, extent$	FeatureBagging	`pyod`	α	MDS	`sklrn`	$p = 2$
COF	`pyod`	k, α	IsoFor	`pyod`	α	KPCA	`sklrn`	$p = 2, kernel$
CBLOF	`pyod`	k, α	COPOD	`pyod`	α	TSNE	`sklrn`	$p = 2$
EllEnv	`sklrn`	α	MCD	`pyod`	α	RSP	`sklrn`	$p = 2$
OCSVM	`sklrn`	α, ν	HBOS	`pyod`	α	LLE	`sklrn`	$p = 2, method$
						SE	`sklrn`	$p = 2$

4.1.3 Imbalanced learning competitors

In order to establish to which extent FROID is in line with state-of-the-art approaches for solving the imbalanced learning problem, we compared the performance of FROID against the following *competitors*.[13] Besides standard approaches for imbalanced learning such as Random Undersampling (RUND) (Kubat & Matwin, 1997), Random Oversampling (ROVE) (Kubat & Matwin, 1997), Synthetic Minority Oversampling Technique (SMOTE) (Chawla et al., 2002), and Adaptive Synthetic (ADASYN) (He et al., 2008), we compared FROID also against ClUstered REsampling (CURE) (Bellinger et al., 2019), Radial-Based Oversampling (Koziarski et al., 2019) (RBO), Combines Cleaning and Resampling (CCR) (Koziarski & Wozniak, 2017), SVMSMOTE (SVMSMT) (Nguyen et al., 2011), and Sampling WIth the Majority (SWIM) (Sharma et al., 2018). Finally, we adopted and re-implemented the eXtreme Gradient Boosting Outlier Detector (XGBOD) (Zhao & Hryniewicki, 2018) for the tasks of both imbalanced learning and supervised outlier detection for which the algorithm is designed.[14]

[13] Unfortunately, we were not able to compare FROID against the following methods either because their code was not publicly available or because it was available in other not comparable languages such as Matlab, C#, and old Python versions: ODBOT, GDLD.

[14] We highlight that, for FROID we selected the same OD method applied by XGBOD (Zhao & Hryniewicki, 2018) - i.e., kNN with all methods and k, IsoFor with different n, LoOP with all k, LOF and LoOP with multiple k neighbors, OCSVM with different ν.

4.1.4 Evaluation measures

As evaluation measures, we considered the following metrics (Tan, 2005). *Precision* is the fraction of relevant instances among the retrieved instances, while *Recall*, also known as *True Positive Rate* or *Sensitivity*, is the fraction of relevant instances that were retrieved, i.e., $Precision = \frac{tp}{tp+fp}$ and $Recall = TPR = Sensitivity = \frac{tp}{tp+fn}$ where tp is the number of true positives, fp is the number of false positives, and fn is the number of false negatives. The *F1-measure* is the harmonic mean of *Precision* and *Recall*, i.e., $F1 = 2\frac{Precision \cdot Recall}{Precision+Recall}$. Another widely used operator to judge the performance of ML classifier is the Area Under the ROC Curve *AUC*, i.e., the area under the curve described by *FPR* and *TPR*, where $FPR = \frac{fp}{fp+tn}$. An index typically used to evaluate the performance of credit score models (Torrent et al., 2020) is the *GINI* coefficient defined as $GINI = 2AUC - 1$. It ranges from 0 (chance results) to 1.0, which corresponds to perfect discrimination between classes. The Precision-Recall Area Under the Curve *PRA* is typically used to judge the performance of ML models on heavily imbalanced datasets because it cares less about the major negative class (Saito & Rehmsmeier, 2015). The *PRA* curve and can be viewed as the average of *Precision* calculated for each *Recall* threshold. We highlight that all the aforementioned metrics are designed to evaluate binary classifiers with respect to the positive class. Hence, in our experiments, we report the average score obtained considering every class of the various datasets analyzed as positive if not differently specified. Finally, we also evaluated the Geometric mean (*GM*) as the root of the product of class-wise *Sensitivity*. This measure tries to maximize the accuracy on each of the classes while keeping these accuracies balanced. We judge the classification results with this long list of measures in order to assess the goodness of the various pre-processing techniques with respect to different and complementary objective evaluation perspectives.[15] For all measures, the higher the values, the better the results. In the rest of the paper, we report aggregated results in terms of average score of the evaluation metric, average ranking position w.r.t. a certain evaluation metric, and number of wins. Detailed results on the various datasets can be found in the Appendix. We base our observations mainly on *PRA* as it is the evaluation measure most widely used for assessing the goodness of imbalanced learning tasks.

Furthermore, we evaluated the time required by the various pre-processing competitors and FROID versions to prepare the dataset (*P-Time*), and its subsequent impact on the training *T-Time*.

4.2 Performance analysis of different ML models

In Tables 12, 13, 14, 15 in the Appendix we show the detailed comparison of the performance among different ML models[16] w.r.t. *PRA* by comparing the performance obtained on the original data X with those obtained on the pre-processing returned by FROID indicated with X_a. Table 3 summarizes these results for PRA, GINI, GM, and F1 by reporting the average scores, average ranks, and the number of wins (between considering X_a and not

[15] We adopted the `sklearn` and `imblearn` libraries to calculate such scores, or we re-implemented the evaluation measures when necessary.

[16] For every table, we show the scores up to three-digit after the comma. Best performers are highlighted considering all the digits that are not shown for space reasons.

considering X FROID preprocessing).[17] We notice that for PRA, LightGBM achieves the best performance overall and significantly overcomes the model not using FROID. Besides *PRA*, LightGBM with FROID is ranked second for *GINI*, and *F1*. We also observe how CatBoost and Random Forest do not benefit from the usage of FROID reporting the overall best performance w.r.t *GINI* and *GM*. Since we consider *PRA* as the most reliable indicator for the class imbalance setting and in order to avoid the repetition of results due to multiple ML models, if not differently specified, in the rest of the paper, we only report the performance related to the LightGBM classifier that we assume to be the best ML model for the datasets analyzed and also because it is notoriously faster than XGBoost (Ke et al., 2017). The non-parametric Friedman test that compares the average ranks of learning methods over multiple datasets w.r.t. the various evaluation measures guarantees that these results are statistically significant, i.e., the null hypothesis that all methods are equivalent is rejected ($p\text{-}value < .001$). This result is verified for every table presented in this paper. Thus, we avoid repeating the statistical significance of the experiments in the following sections.

4.3 Pre-processing with different features combinations

In this section, we analyze the impact of the different features adopted by FROID. Table 4 reports the average *PRA*, *GINI*, *F1*, *GM*, the corresponding average ranks, and number of wins among all datasets classified with LightGBM on the various datasets analyzed for different pre-processing inputs:[18]

- X is the original dataset,
- X_a is the output of FROID,
- $X_{\neg_\zeta}$ is not using the features selection function ζ,
- X_σ is using variance threshold as feature selection ζ with threshold set as.2,
- X_f is not using the original features of X, but only all the unsupervised features learned by FROID,
- X_l is considering X_o and X_r besides X,
- X_{fo} is only considering X_o, X_{oo}, X_{or},
- X_{fr} is only considering X_r, X_{rr}, and X_{ro},
- X_O is considering X, and X_{fo},
- X_R is considering X and X_{fr},

where X_o, X_r, X_{oo}, X_{rr}, X_{ro}, X_{or} are defined as in Sect. 3. What emerges is that FROID is firmly ranked first w.r.t. the four measures, it has the best performance w.r.t. *GM* and *F1* and it is runner up for *PRA* and *GINI*. Also regarding the number of wins is always placed

[17] For each of the different classifiers, we also performed a pairwise Wilcoxon test to assess whether there is a statistically significant difference in performance between FROID and the classifier trained on the original dataset. For the Decision Tree, the Random Forest and CatBoost we do not observe statistically significant differences, while for LightGBM and XGBoost we have statistically significant results for PRA and Gini at .05 significance level.

[18] We tested these semantically-meaningful combinations because *(i)* testing all the possible randomly selected combinations would have led to an explosion of the total number of possibilities, and *(ii)* we did not remove the features one of them after the other because their individual absence is not impactful for the performance.

Table 3 Average scores, ranks and number of wins of *PRA*, *GINI*, *F1*, *GM* among various datasets and ML models trained on original data X or on data pre-processed with FROID X_a

		Decision tree		CatBoost		LightGBM		Random forest		XGBoost	
		X	X_a	X	X_a	X	X_a	X	X_a	X	X_a
PRA	avg	.435	.437	**.693**	*.683*	.626	.675	.655	.644	.637	**.693**
	rank	1.39	1.39	1.44	1.38	1.53	**1.30**	*1.34*	1.50	1.50	*1.34*
	wins	39/64	39/64	36/64	40/64	30/64	**45/64**	42/64	32/64	32/64	*42/64*
GINI	avg	.513	.526	**.842**	*.834*	.805	.832	.809	.806	.792	.825
	rank	1.42	1.44	**1.28**	1.53	1.53	*1.30*	1.41	1.44	1.47	1.38
	wins	37/64	36/64	**46/64**	30/64	30/64	*45/64*	38/64	36/64	34/64	40/64
GM	avg	.647	*.659*	.636	.602	.604	.639	.564	.587	.624	**.660**
	rank	1.36	1.42	**1.28**	1.41	1.39	1.36	**1.28**	1.39	*1.34*	1.38
	wins	41/64	37/64	**46/64**	38/64	39/64	41/64	**46/64**	39/64	*42/64*	40/64
F1	avg	.530	.536	*.587*	.556	.546	.584	.520	.531	.563	**.607**
	rank	1.41	1.38	1.28	1.41	1.41	*1.34*	**1.27**	1.41	1.36	1.36
	wins	38/64	40/64	46/64	38/64	38/64	*42/64*	**47/64**	38/64	41/64	41/64

The rank and wins are calculated for the couples X and X_a for each ML model. Best performer in bold

second. The overall champion regarding the number of dataset for which it is first is the LightGBM on the original data X. However, the performance for the remaining datasets place it among the last positions w.r.t. to the rank indicator, and constantly and statistically worse than various alternatives using FROID.

From the comparison between X_a, $X_{\neg\zeta}$ and X_σ emerges that *(i)* the usage of a feature selection method contributes in increasing the performance, and *(ii)* that the usage of more efficient but less adaptive feature selection functions like variance threshold impact negatively the performance of the model. Thus, the impact of the class imbalance on the model used by the feature selection method, is the aspect that contributes in the increase of the performance and the appropriate usage of the most reliable set of features.

Detailed performance on the various datasets are available in Tables 16, 17, 18, 19 in the Appendix. Here, among the other evaluation measures, we can observe that, in many cases, FROID boosts the PRA on the original dataset with an improvement ranging from.1% to 81.2%, with an average boost of 12%. Therefore, this experiment confirms that it makes sense to consider all the alternative unsupervised features created by FROID together with the original ones and appropriately selected and not only a subset of them.

Furthermore, in Table 5 we report the average features importance obtained by FROID (X_a) over all the datasets paired with the rank of the features importance[19] w.r.t. the different categories of unsupervised features adopted. We notice that the features involving FR not mixed with OD, i.e., X_r and X_{rr} are, on average, the most beneficial for the classification, followed by single OD X_o and by the original features X. However, we notice that there is no marked discrepancy in the usage of the features and that the boost of FROID is given by the simultaneous usage of all the categories of features created. The detailed relative importance of the various datasets is available in Table 20 in the Appendix. In Table 6, we report the average relative features importance grouped by category of features

[19] We remark that the rank in Table 5 is related to the features importance and not to the performance.

Table 4 Average scores, ranks and number of wins of *PRA*, *GINI*, *F1*, *GM* among various datasets obtained by LightGBM trained on different features combinations

		X	X_a	$X_{\neg\zeta}$	X_σ	X_f	X_l	X_{fo}	X_{fr}	X_O	X_R
PRA	avg	.626	*.675*	**.676**	.670	.642	.671	.553	.628	.663	.667
	rank	5.08	**3.67**	3.83	4.17	4.56	4.56	7.44	5.31	4.86	3.75
	wins	**23/64**	*18/64*	*18/64*	17/64	15/64	16/64	10/64	*18/64*	*18/64*	17/64
GINI	avg	.805	*.832*	**.834**	.825	.803	.815	.739	.801	.811	.815
	rank	4.77	**3.34**	*3.64*	4.06	5.21	4.61	7.28	5.31	4.69	4.35
	wins	**20/64**	18/64	**20/64**	18/64	12/64	18/64	11/64	16/64	18/64	*19/64*
GM	avg	.604	**.639**	*.638*	.634	.610	*.638*	.494	.610	.619	.641
	rank	4.09	**3.19**	*3.42*	3.72	3.95	3.84	6.52	4.66	4.39	3.22
	wins	**27/64**	22/64	24/64	25/64	22/64	23/64	14/64	18/64	19/64	*26/64*
*F*1	avg	.546	**.585**	.584	.581	.550	.578	.435	.542	.564	.581
	rank	4.09	**3.19**	3.33	3.52	3.87	4.00	6.56	4.75	4.42	3.27
	wins	**26/64**	*23/64*	*23/64*	**26/64**	22/64	*23/64*	14/64	18/64	19/64	**26/64**

Best performer in bold

Table 5 Average relative importance and ranks of importances by category of features for FROID: X original features, X_o OD over original, X_r FR over original, X_{rr} FR over FR, X_{ro} OD over FR, X_{oo} OD over OD, X_{rr} FR over FR

	X	X_o	X_r	X_{rr}	X_{ro}	X_{or}	X_{oo}
Rank	3.156	3.141	3.195	2.719	5.453	4.789	5.547
Avg	0.256	0.192	0.182	0.226	0.045	0.063	0.037

for FROID for the clusters of similar datasets described in Sect. 4.1. The insights of this table are the following: for datasets in cluster A, we have the general behavior already discussed for Table 5. The original features are consistently more important for datasets in cluster D, while for datasets in cluster B are beneficial features in X_{rr}. Finally, for datasets in cluster C, the features in X_{oo} are not used at all, while are more important those in X_{ro}. Hence, we can infer that the effectiveness of FROID is given by the massive production of unsupervised descriptive features that can be helpful in every situation, independently from the dataset characteristics. This improvement can be effectively exploited only by ML models like LightGBM that can appropriately select the most discriminative and informative features and are not harmed by the course of dimensionality issue.

Finally, we studied if there are sets of features in X_o, X_{oo}, X_{or}, X_r, X_{rr}, and X_{ro} that are never or scarcely used by the classifiers adopted. An analysis performed at the dataset level highlighted that for all the analyzed datasets, FROID uses original features X in $\sim$ 97% of the datasets, while the features generated by FROID, i.e., X_o, X_r, X_{rr}, X_{ro}, X_{or}, and X_{oo} are instead used in $\sim$ 92%, $\sim$ 84%, $\sim$ 87%, $\sim$ 80%, $\sim$ 80% and $\sim$ 71% of the datasets, respectively. Thus, all the types of features are consistently used in more than half of the datasets analyzed. This result emphasizes how FROID self-adapts promptly to each dataset's peculiarities.

Table 6 Average relative features importance grouped by category of features for FROID for different clusters of similar datasets

Cluster	X	X_o	X_r	X_{rr}	X_{ro}	X_{or}	X_{oo}
A	.241	.205	.190	.212	.042	.067	.040
B	.250	.024	.071	.600	.028	.012	.012
C	.233	.042	.107	.299	.311	.005	.000
D	.469	.155	.168	.116	.027	.039	.023

4.4 Comparison with state-of-the-art approaches

In this section we compare FROID against state-of-the-art approaches for imbalanced learning. In Table 7, are reported the average *PRA*, *GINI*, *F1*, *GM*, the corresponding average ranks among all datasets classified with LightGBM for the various competitors and the number of wins. What emerges is that, overall, FROID is the best pre-processing method with respect to the four evaluation measures. Also, there is not a clear second-best performer, even though SVMSMT and SWIM are the best ones for some indicators. Thus, FROID appears to be markedly better than the other approaches. The comparison of the ranks of all methods against each other is visually represented in Fig. 3 with Critical Difference (CD) diagrams (Demsar, 2006). Two methods are tied if the null hypothesis that their performance is the same cannot be rejected using the Nemenyi test at $\alpha = .05$. For PRA, FROID is the only pre-processing method not tied with other approaches ranked less than seventh. This means that w.r.t. PRA, it is statistically insignificant to use FROID or other sate-of-the-art approaches like SVMSMT, SWIM or RBO. Furthermore, even though the results show that FROID is always statistically tied with other methods, independently from the evaluation measure considered, it has always the highest number of wins, it is in the top three with respect to the rank for PRA, GM, and F1, and it has the highest average PRA and GINI. No other method guarantees such stability across different evaluation measures. To further enhance the improvement of FROID vs `orig`, FROID vs SVMSMT, and FROID vs SWIM w.r.t *PRA* we report in Fig. 4 scatter plots in which every point represent a dataset and along the x-axis and y-axis we have the performance in terms of *PRA* for the method reported in the label. The closer the point is to the diagonal, the more similar the performance. The leftmost scatter plot highlights that only in a few cases not using FROID is better than using it and that, in some cases, its usage brings a considerable boost in terms of *PRA*. The central scatter plot signals that FROID is never markedly worse than SVMSMT as all the points below the diagonal are close to it, but in some cases, the performance of FROID are markedly better than those of SVMSMT. The rightmost scatter plot highlights how the performance of FROID are correlated to those of SWIM but the majority of the points lies above the diagonal signaling the superiority of our proposal.

The drawback of using FROID is the markedly higher time required to prepare the dataset. Indeed, while the training time T remains in the same order of magnitude for all the methods (except XGBOD and RBO), the pre-processing time P becomes consistently higher, being in the order of hours instead of in the order of seconds for FROID. However, we underline that the standard deviation of the time required by FROID is 6,436.18 while the median time is 231.22 s. The great variability in the pre-processing time P indicates that FROID pre-processing time is markedly impacted by the dimensionality of the dataset analyzed. We recall that FROID is designed as a pre-processing method to be used when a good and reliable model should be deployed, and the time required to build this model is

Table 7 Average scores, average ranks and number of wins of *PRA*, *GINI*, *F1*, *GM*

		ORIG	FROID	XGBOD	ROVE	RUND	CURE	RBO	CCR	ADASYN	SMOTE	SVMSMT	SWIM
PRA	avg	.626	**.675**	.673	.641	.368	.619	.653	.651	.643	.653	.674	.668
	Rank	6.33	**5.01**	6.46	6.62	10.45	7.61	5.95	5.87	6.60	6.29	5.13	5.32
	Wins	14/64	**24/64**	15/64	12/64	5/64	14/64	15/64	17/64	12/64	12/64	*19/64*	18/64
GINI	avg	.805	**.832**	.818	.779	.534	.794	.810	.812	.805	.817	.827	**.832**
	Rank	6.34	5.41	6.84	7.45	10.05	7.39	6.07	5.35	6.60	5.98	5.29	**4.87**
	Wins	13/64	**22/64**	15/64	12/64	6/64	11/64	18/64	15/64	13/64	16/64	16/64	*20/64*
GM	avg	.604	.639	**.655**	.594	.392	.652	.571	.548	.582	.612	.615	.559
	Rank	5.70	5.34	5.61	6.29	7.53	**5.30**	7.23	8.01	6.85	6.13	6.03	7.63
	Wins	14/64	**24/64**	19/64	13/64	17/64	*23/64*	15/64	13/64	8/64	15/64	17/64	15/64
*F*1	avg	.546	.585	**.596**	.546	.308	.549	.527	.495	.521	.555	.553	.514
	Rank	5.45	**5.16**	5.48	5.93	8.91	6.09	6.82	7.66	6.76	5.98	5.84	7.55
	Wins	18/64	**24/64**	*21/64*	18/64	11/64	18/64	17/64	13/64	9/64	15/64	18/64	15/64
P	Avg	–	2,359	558.28	5.56	3.21	14.36	3,627	24.57	6.25	6.12	6.01	142.11
T	Avg	.09	.73	.51	1.28	.43	.78	4.84	1.38	.79	.95	.69	1.56

Average pre-processing (P) and training (T) times among various datasets for LightGBM trained after different methods. Best performer in bold, runner up in italic

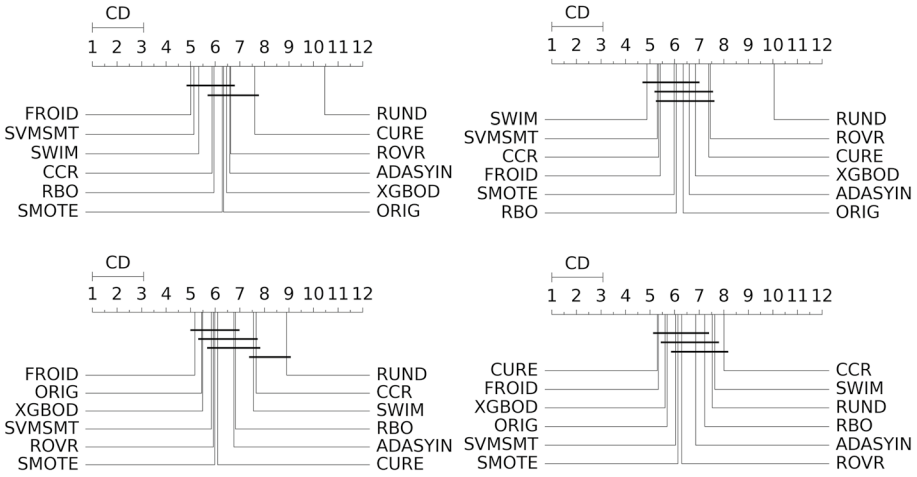

Fig. 3 Critical difference plots with Nemenyi at 95% confidence level for PRA (top-left), GINI (top right), F1 (bottom-left), GM (bottom-right)

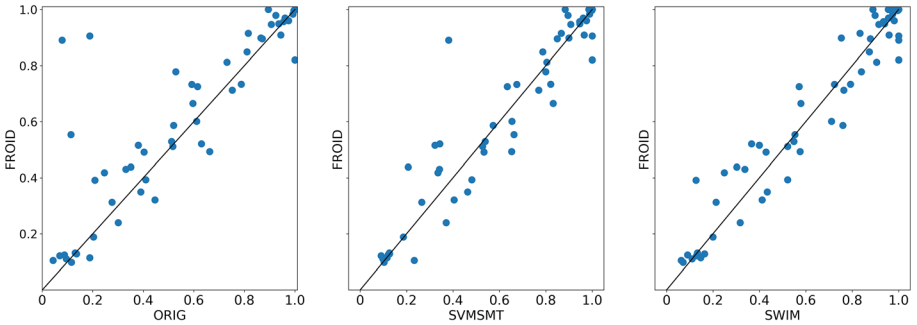

Fig. 4 Scatter plots comparing the performance in terms of *PRA* between couples of methods. Every point is a dataset, the closer is to the diagonal, the more similar the performance

Table 8 Average scores, average ranks and wins of *PRA*, *GINI*, *F1*, *GM* for LightGBM trained after different pre-processing methods with/without hyper-parameter tuning (-HPT) or calibration (-CAL)

		ORIG	ORIG-CAL	ORIG-HT	FROID	FROID -CAL	FROID -HT
PRA	Avg	.626	.633	.610	.675	**.678**	.657
	Rank	3.61	3.43	4.21	3.05	**2.91**	3.73
	Wins	16/64	21/64	15/64	*22/64*	**25/64**	20/64
GINI	avg	.805	.797	.774	**.832**	.824	.798
	Rank	3.44	3.62	4.26	**2.85**	2.96	3.83
	Wins	18/64	19/64	14/64	*24/64*	**25/64**	19/64
GM	avg	.604	.567	.001	.639	**.644**	.000
	Rank	2.68	2.96	5.11	2.63	**2.44**	5.12
	Wins	**33/64**	26/64	7/64	27/64	**33/64**	7/64
*F*1	avg	.546	.523	.015	.585	**.589**	.014
	Rank	2.66	2.99	5.11	2.55	**2.50**	5.12
	Wins	**33/64**	25/64	7/64	29/64	*32/64*	7/64

Best performer in bold, best performer runner up in italic

typically not an issue. Details of the performance on the various datasets are available in Tables 21, 22, 23, 24 in the Appendix. In order to recover from the high computational time required by the FROID framework, we have developed a streamlined variant that focuses solely on the most efficient OD and FR methods. Compared to the original FROID, this lightweight version demonstrates a remarkable performance boost, being approximately an order of magnitude faster. However, its effectiveness in addressing the imbalanced learning problem falls short in terms of PRA and GINI when compared to both FROID and other state-of-the-art approaches. The average rank for the PRA and GINI metrics was approximately 10th and 9th, respectively. Regrettably, due to the limitations of this lightweight version in achieving comparable accuracy without utilizing the full range of OD and FR methods employed by FROID, we have made the decision not to include it in our experimentation. We believe that the complete FROID framework remains the most reliable option for achieving optimal performance in solving the imbalanced learning problem. Therefore, we have chosen not to report the results obtained using the lightweight version.

Furthermore, we analyzed which is the impact of procedures of *hyper-parameter* tuning and *calibration* (Niculescu-Mizil & Caruana, 2005; Zadrozny & Elkan, 2002) on the performance of the LightGBM for the various datasets.[20] We use the suffix -HPT and -CAL to indicate a training procedure involving hyper-parameter tuning or calibration, respectively. Table 8 illustrates the average *PRA*, *GINI*, *F1*, *GM*, the corresponding average ranks and number of wins among all datasets classified with LightGBM on the original dataset (ORIG), after FROID pre-processing with and without hyper-parameter tuning (-HPT) or calibration (-CAL). The results show that hyper-parameter or calibration approaches alone do not reach the performance achieved by FROID. Also, calibration on top of the models trained after FROID pre-processing further improves the performance.

4.5 Results on supervised outlier detection

In this section, we demonstrate that the pre-processing of FROID is beneficial not only for imbalanced learning but also for supervised outlier detection. As competitors, we report the same approaches used in the previous section, with XGBOD being the actual state-of-the-art in this field (Zhao & Hryniewicki, 2018). Table 9 reports the PRA w.r.t. the label "is outlier" on the datasets having the ground truth for the outliers. We observe that FROID is the best performer for `satellite` and second best performer for `glass`. This result further stress the breakthrough idea we introduced in this paper about representing instances towards a variegate composition of unsupervised OD and FR approaches.

4.6 Features importance on real case studies

In this section, we experimented with the effectiveness of FROID in two real case studies.[21] In particular, the `diva` dataset is a privately released dataset on fraud evasion, periodically issued by the Italian Ministry of Economics. In `diva`, financial activities for 11,187

[20] For calibration we employed a 5-fold cross calibration classifier as implemented in `scikit-learn` using as a base classifier a LightGBM. For tuning, we employed a 5-fold grid search CV as implemented in `scikit-learn`, using LightGBM. The parameter space we tested is the following: $learning\_rate : [.01, .1, 1], n\_estimators : [50, 100, 200], reg\_alpha : [0, 1e-1, 1], reg\_lambda : [0, 1e-1, 1]$. We selected the best performing combination w.r.t AUC.

[21] Due to privacy and legal concerns, we are not allowed to publicly release these datasets.

Table 9 PRA for supervised outlier detection datasets obtained by LightGBM trained after different preprocessing methods.

dataset	ORIG	FROID	XGBOD	ROVE	RUND	CURE	RBO	CCR	ADASYN	SMOTE	SVMSMT	SWIM
ecoli	.247	.503	.515	.698	.034	.555	*.542*	.699	.582	.550	**.705**	.698
glass	.629	*.828*	.316	.573	.048	.335	.573	**.840**	.520	.491	.397	.573
satellite	.968	**.970**	.964	.966	.963	.964	.967	*.970*	.964	.968	.968	.967

The best approach is highlighted in bold

Table 10 Performance on the case study datasets with and without FROID

	diva		hospital	
	PRA	GINI	PRA	GINI
ORIG	.41	.90	.24	.83
FROID	.42	.97	.27	.84
↑	2.44%	7.78%	12.5%	1.20%

citizens are recorded. The 18 features describe different aspects of the taxpayers, including their past financial credit score, declared income and property value, debt, and detailed taxation info. The positive label mark 35 relevant tax evaders, accounting for less than .3% of the total labels. The hospital dataset collects 14,390 records describing patients through 15 features after a data cleaning phase, including demographic variables, hospital usage aspects, and past medical history. The positives indicate the 130 discharged patients, accounting for roughly .9% of the patients. We applied to these datasets the same preprocessing described in Sect. 4.1. The comparison of the performance between the LightGBM obtained on the original dataset and those obtained after running FROID are reported in Table 10. The last line reports the relative improvement of FROID over the performance on the original datataset. We observe how by using FROID, we can correctly identify many more fraudulent citizens and discharged patients. Indeed, the GINI[22] obtained b^y FROID increases up to .97 and .84, an increase by 7.8% and 1.2% respectively compared to ORIG.

In Fig. 5 is reported the normalized features importance of the ten most important features for both datasets. We immediately notice that for both cases, among the ten most important features, we have many features generated by FROID. In particular, we notice how using OD approaches is quite an effective procedure to design highly discriminative features over the diva dataset. Moreover, for hospital, we see beneficial effects of FR methods: *KPCA* is among the top 10 most informative features when applied not only on original data but also on scores derived from OD and FR methods. This analysis confirms that FROID's idea of making inception among FR and OD methods (and vice-versa) is indeed a winning one as it helps to derive the most discriminative and important features for the classification in the setting of imbalanced learning. We highlight from this analysis also emerging weaknesses w.r.t. the usage of FROID. Indeed, while the overall performance improves, the usage of FROID leads to an inevitable loss of interpretability (Guidotti et al.,

[22] GINI is the main measure required by the stakeholders involved.

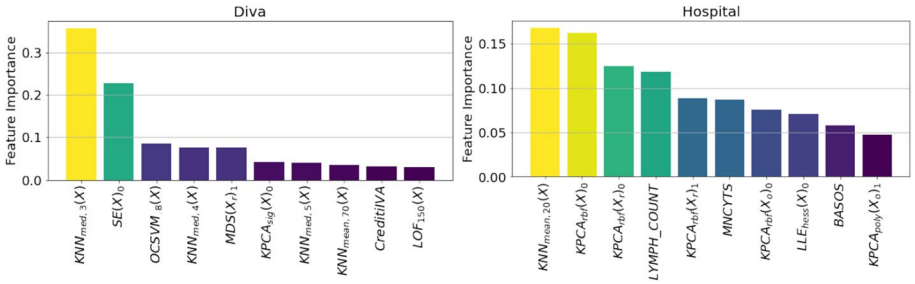

Fig. 5 Normalized LightGM features importance for `diva` (left) and `hospital` (right)

2018). In fact, if the original dataset is used the features used by the classifier selected are only those belonging to the original domain, and therefore their meaning is perfectly understandable by any domain expert. On the other hand, when features created by FROID becomes fundamental for the classification, only machine learning experts can understand their meaning (Tomsett et al., 2018). On the contrary, this weakness is not held by classic approaches like SMOTE or ADASYN, which do not modify the features describing the dataset.

5 Conclusion

We have presented FROID an unsupervised pre-processing framework for solving imbalanced learning problems through outlier detection and features reduction methods. FROID augments the dimensions used to represent the input data combining in different ways a wide variety of outlier detection and dimensionality reduction methods. This dimensionality augmentation boosts ML classification models in solving the classification task for imbalanced data. A wide and deep experimentation shows that FROID overcomes state-of-the-art pre-processing approaches for imbalanced learning at the price of a not negligible time required to build all the novel dimensions. Our insight for such boost in performance is that FROID does not generate any novel synthetic data but only amplify the expressiveness of the existing records. The effectiveness of FROID in its current form is that it is creating a mass of descriptive discriminative features that can be suitable to any dataset, i.e., maybe a subset might not be helpful for datasets with specific characteristics, but they might be useful for other datasets. Indeed, we might see FROID as a sort of "brute-force approach" generating a massive number of descriptive features in the hope that one of them, or better, a combination of some of them, together with the original feature, bring a boost to the discriminative power of a ML model.

Starting from the results obtained with FROID, several future research directions can be pursued. First, we would like to design a pre-processing step to be applied before FROID that is responsible for which is the most appropriate subset of features to generate

while simultaneously improving the performance and reducing the running time. Second, inspired by Shi et al. (2020); Ibrahim (2021), we would like to experiment which would be the performance obtained by combining FROID with one of the state-of-the-art oversampling and/or undersampling approaches as well as with cost-sensitive classifiers. For instance, if SMOTE is applied after FROID, then it will generate synthetic minority instances approximating the features learned by FROID. On the one hand, this might boost the discriminative power of ML models learned on top of these datasets enriched both in terms of features and in terms of records. However, on the other hand, there is the risk that SMOTE would not be able to appropriately generate synthetic instances with the features learned by FROID, leading to a degradation of the performance. Third, we would like to design an extremely randomized version of FROID that adopts bootstrap samples and random features selection to simultaneously accelerate the procedure and also exploit a sort of ensemble strategy. Finally, we would like to check if the FROID approach can be used to solve the imbalanced learning problem also for multi-class problem settings and for other data types such as images or time series through the usage of autoencoder approaches.

Appendix A: Additional results

For the sake of completeness, we report in this section the detailed experimental results obtained. Tables are placed after References. (See Tables 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24).

Table 11 Datasets description: n_{train} instances training set; n_{test} instances test set; m number of features; m_{num} number numerical features; p^{+}_{train} positive rate training; p^{+}_{test} positive rate test set; *FDR* Maximum Fisher's Discriminant Ratio, *FBP* Fraction of Borderline Points, *ECP* Entropy of Class Proportions and *IR* Imbalance Ratio

Dataset	n_{train}	n_{test}	m	m_{num}	p^{+}_{train}	p^{+}_{test}	*FDR*	*FBP*	*ECP*	*IR*
abalone1	2921	1253	8	7	.008	.008	.994	.022	.936	.985
abalone2	1135	487	8	7	.019	.021	.994	.048	.862	.960
abalone3	1341	575	8	7	.013	.014	.955	.028	.897	.973
abalone4	406	175	8	7	.025	.023	.795	.044	.833	.950
abalone5	351	151	8	7	.028	.033	.564	.011	.813	.941
abalone6	511	220	8	7	.057	.059	.875	.114	.686	.880
arrhythmia	316	136	278	278	.054	.059	.874	.149	.698	.887
avila	10430	10437	10	10	.010	.010	.994	.011	.920	.980
bank	3164	1357	42	7	.115	.115	.840	.216	.484	.744
careval1	1209	519	21	21	.078	.077	.930	.100	.606	.833
careval2	1209	519	21	21	.037	.039	.923	.076	.771	.923
cardio	25914	11107	11	11	.054	.054	.984	.142	.697	.886
coil1	6875	2947	85	85	.060	.060	.979	.152	.674	.874
covtype	26950	11551	10	10	.071	.071	.950	.037	.629	.847
derma1	250	108	34	0	.056	.056	.087	.000	.689	.882
drybean	9527	4084	16	16	.038	.038	.329	.000	.766	.920
ecoli1	235	101	7	7	.085	.089	.812	.077	.580	.816
ecoli2	232	100	6	6	.073	.080	.820	.047	.622	.843
ecoli3	235	101	7	7	.060	.059	.748	.038	.674	.874
isolet	5457	2340	617	617	.077	.077	.954	.065	.609	.834
kddcup1	1563	670	38	15	.013	.013	.191	.001	.897	.973
kddcup2	1149	493	38	15	.032	.032	.115	.003	.795	.934
kddcup3	742	319	38	15	.020	.019	.184	.003	.857	.959
kddcup4	1127	483	38	15	.013	.012	.200	.002	.898	.973
kddcup5	1557	668	38	15	.010	.010	.241	.003	.922	.981
led7digit1	310	133	7	7	.084	.083	.846	.126	.584	.818
letter	14000	6000	16	16	.037	.037	.882	.006	.773	.924
libras	252	108	90	90	.067	.065	.916	.067	.644	.856
machine	7000	3000	7	5	.034	.034	.965	.053	.787	.930
mgraphy	7828	3355	6	6	.023	.023	.776	.028	.841	.952
oil	655	282	49	49	.044	.043	.812	.063	.738	.908
ozone	1775	761	72	72	.029	.029	.968	.061	.812	.941
page1	330	142	10	4	.061	.056	.580	.030	.670	.872
poker1	1033	444	10	0	.012	.011	.999	.000	.909	.976
poker2	1452	623	10	0	.012	.013	.996	.000	.908	.976
poker3	1039	446	10	0	.016	.018	.997	.000	.880	.967
scene	1684	723	294	294	.074	.073	.956	.160	.621	.842
shuttle1	2321	995	9	0	.015	.015	.406	.000	.890	.970
shuttle2	1280	549	9	0	.067	.067	.273	.000	.645	.857
solar1	972	417	32	32	.049	.048	.928	.086	.716	.896
spect	371	160	93	93	.084	.088	.752	.049	.585	.819
stars	579	249	3	3	.862	.859	.480	.059	.421	.687

Table 11 (continued)

Dataset	n_{train}	n_{test}	m	m_{num}	p^{+}_{train}	p^{+}_{test}	*FDR*	*FBP*	*ECP*	*IR*
thyroid	2640	1132	52	52	.061	.061	.916	.089	.667	.870
uscrime	1395	599	100	100	.075	.075	.736	.127	.615	.838
vowel	691	297	13	10	.091	.091	.756	.033	.560	.801
webpage	24346	10434	300	300	.028	.028	.930	.035	.815	.942
wine1	483	208	11	11	.014	.014	.952	.027	.891	.971
wine2	1119	480	11	11	.033	.033	.967	.085	.790	.932
wine3	598	257	11	11	.022	.019	.977	.059	.849	.956
wine4	630	270	11	11	.022	.022	.948	.040	.846	.955
yeast1	1038	446	8	8	.035	.034	.916	.066	.783	.928
yeast2	1691	726	103	103	.074	.073	.994	.182	.620	.841
yeast3	702	302	8	8	.098	.099	.876	.134	.536	.785
yeast4	702	302	8	8	.098	.099	.769	.085	.536	.785
yeast5	354	152	8	8	.099	.099	.860	.184	.535	.783
yeast6	369	159	8	8	.098	.094	.818	.173	.539	.786
yeast7	321	138	7	7	.065	.065	.948	.137	.651	.861
yeast8	662	285	8	8	.032	.032	.967	.077	.797	.935
yeast9	485	208	8	8	.043	.043	.982	.105	.743	.910
yeast10	359	155	8	8	.100	.097	.715	.097	.530	.780
yeast11	337	145	8	8	.042	.041	.489	.047	.751	.913
yeast12	1038	446	8	8	.035	.034	.905	.074	.783	.928
yeast13	1038	446	8	8	.030	.029	.862	.037	.806	.938
yeast14	1038	446	8	8	.023	.025	.928	.046	.841	.953

Table 12 PRA among various datasets and ML models trained on original data X or on data pre-processed with FRODO X_a

Dataset	Decision tree		CatBoost		LightGBM		Random forest		XGBoost	
	X	X_a	X	X_a	X	X_a	X	X_a	X	X_a
abalone1	*.028*	.021	.060	*.085*	*.115*	.098	.050	*.106*	**.146**	.134
abalone2	*.062*	.039	.388	*.569*	**.630**	.521	.422	*.436*	.438	*.610*
abalone3	.108	*.363*	*.695*	.684	.596	*.665*	*.567*	.534	.495	**.885**
abalone4	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
abalone5	*.008*	*.008*	.071	*.092*	.043	*.105*	**.135**	.042	*.070*	.058
abalone6	*.080*	.075	.420	*.540*	.380	*.515*	.395	*.489*	.377	**.556**
arrhythmia	.645	*.789*	.919	**.988**	.924	*.979*	.649	**.988**	.941	*.988*
avila	*.970*	.883	**.997**	.972	*.992*	.983	*.984*	.811	*.989*	.964
bank	.222	*.238*	.516	*.523*	*.517*	.511	*.490*	.436	**.540**	.511
careval1	*.743*	.314	*.949*	.914	*.944*	.909	*.932*	.636	**.962**	.911
careval2	*.952*	.244	**1.00**	.833	**1.00**	.820	*.981*	.823	**1.00**	.765
cardio	.071	*.078*	*.197*	.191	**.203**	.188	*.181*	.169	*.188*	.168
coil1	.073	*.075*	.126	*.132*	.130	*.132*	*.122*	.119	**.133**	.132
covtype	*.713*	.649	*.964*	.957	.955	*.957*	*.942*	.934	**.964**	.955
derma1	.612	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
drybean1	*.994*	*.994*	**1.00**	**1.00**	.999	**1.00**	**1.00**	**1.00**	.994	**1.00**
ecoli1	.491	*.718*	.853	*.882*	.870	*.895*	.845	*.849*	.877	**.901**
ecoli2	.229	*.575*	.805	**.821**	.528	*.778*	.752	*.803*	.630	*.809*
ecoli3	*.708*	.589	.958	**1.00**	.936	*.949*	.958	**1.00**	.973	*.982*
isolet	.364	*.665*	.960	**.979**	.961	*.969*	.935	*.971*	.951	*.961*
kddcup1	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
kddcup2	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
kddcup3	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
kddcup4	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
kddcup5	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
led7digit1	*.484*	.199	.650	*.691*	.615	*.725*	.626	*.659*	.624	**.732**
letter	*.905*	.775	*.996*	.995	.996	*.996*	*.995*	.986	**.998**	.995
libras	.274	**1.00**	.985	**1.00**	.895	**1.00**	.914	**1.00**	.877	**1.00**
machine	*.468*	.300	*.791*	.725	*.788*	.734	*.770*	.541	**.814**	.731
mgraphy	*.359*	.288	**.776**	.741	*.752*	.712	*.755*	.681	*.737*	.715
oil	*.157*	.129	.389	*.417*	*.390*	.350	**.437**	.361	*.399*	.370
ozone	.062	*.067*	.262	*.294*	.276	**.313**	.183	*.198*	.233	*.258*
page1	**1.00**	.537	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	.865	**1.00**	**1.00**
poker1	.102	*.123*	.475	**.478**	.350	*.438*	.322	*.395*	.346	*.440*
poker2	.108	*.222*	**1.00**	.911	.078	*.890*	.508	*.854*	.052	*.867*
poker3	*.066*	.012	**1.00**	.858	.188	*.906*	.776	*.864*	.098	*.951*
scene	.092	*.142*	.363	*.421*	.331	*.431*	.283	*.330*	.374	**.431**
shuttle1	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
shuttle2	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
solar1	*.106*	.053	*.159*	.134	*.136*	.128	*.106*	.094	**.184**	.117
spect	.582	*.614*	.877	*.880*	.865	*.898*	*.815*	.785	.865	**.932**
stars	.968	*.977*	**.999**	.998	*.998*	.997	*.998*	.997	*.998*	.997

Table 12 (continued)

Dataset	Decision tree		CatBoost		LightGBM		Random forest		XGBoost	
	X	X_a	X	X_a	X	X_a	X	X_a	X	X_a
thyroid	*.752*	.750	**.981**	.963	*.975*	.960	*.967*	.829	*.971*	.968
uscrime	*.232*	.206	.534	*.550*	.512	*.529*	*.561*	.539	.538	**.580**
vowel	.783	*.865*	**1.00**	**1.00**	**1.00**	**1.00**	.995	**1.00**	.999	**1.00**
webpage	.455	*.487*	**.866**	.861	.810	*.848*	.816	*.829*	.817	*.851*
wine1	*.015*	*.015*	*.483*	.118	.208	*.391*	*.134*	.097	.302	**.706**
wine2	*.032*	*.032*	*.114*	.082	.088	**.125**	*.108*	.044	.064	*.125*
wine3	*.021*	*.021*	.079	**.140**	.069	*.122*	.050	*.078*	.041	*.105*
wine4	*.269*	.197	.691	**.710**	.520	*.586*	*.686*	.666	.560	*.675*
yeast1	*.391*	.216	.649	*.658*	*.611*	.601	**.704**	.696	*.605*	.564
yeast2	.541	*.547*	.914	*.926*	.815	*.915*	**.928**	.900	.836	*.913*
yeast3	.196	*.304*	**.516**	.462	*.410*	.393	*.483*	.454	*.504*	.471
yeast4	.194	*.567*	**.788**	.729	.592	*.733*	.699	*.727*	.755	*.765*
yeast5	*.196*	.115	.386	**.484**	.246	*.418*	*.327*	.327	.283	*.385*
yeast6	*.043*	*.043*	**.240**	.127	*.188*	.114	.121	*.168*	*.115*	.091
yeast7	*.366*	.156	*.595*	.574	.403	*.491*	**.610**	.571	.460	*.565*
yeast8	*.494*	.396	.924	*.925*	.907	**.946**	*.928*	.835	.886	*.908*
yeast9	.090	*.335*	.527	*.639*	.114	*.554*	**.672**	.561	.520	*.583*
yeast10	*.156*	.087	**.533**	.358	*.446*	.321	*.522*	.335	*.436*	.397
yeast11	*.464*	.345	.797	*.800*	.731	*.812*	*.806*	.751	.715	**.896**
yeast12	.183	*.307*	*.650*	.513	*.663*	.493	.574	**.694**	*.648*	.563
yeast13	.084	*.135*	*.353*	.285	*.301*	.240	*.315*	.296	**.374**	.308
yeast14	.072	*.078*	.109	**.123**	.096	*.110*	.075	*.091*	*.101*	.099

Best between X and X_a for each ML model highlighted in italic. Best for each dataset highlighted in bold

Table 13 GINI among various datasets and ML models trained on original data X or on data pre-processed with FROID X_a

Dataset	Decision tree		CatBoost		LightGBM		Random forest		XGBoost	
	X	X_a	X	X_a	X	X_a	X	X_a	X	X_a
abalone1	*.053*	-.023	.323	**.375**	.131	*.348*	.201	*.249*	.206	*.360*
abalone2	*.123*	.102	.903	*.943*	**.968**	.901	*.904*	.820	.884	*.888*
abalone3	.210	*.705*	*.977*	.968	*.967*	.963	.959	*.968*	.864	**.989**
abalone4	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
abalone5	-.004	*-.002*	**.752**	.590	.582	*.647*	*.370*	.347	*.658*	.563
abalone6	.056	*.076*	.583	**.749**	.642	*.735*	.665	*.718*	.509	*.728*
arrhythmia	.623	*.875*	.968	*.998*	.986	*.997*	.932	*.998*	.985	**.998**
avila	*.990*	.850	**1.00**	.998	*1.00*	1.00	*1.00*	.986	*1.00*	.999
bank	.298	*.350*	**.814**	.792	*.801*	.779	*.775*	.732	*.807*	.755
careval1	*.809*	.536	*.992*	.982	*.990*	.988	*.990*	.870	**.994**	.980
careval2	*.998*	.584	**1.00**	.981	**1.00**	.987	*.999*	.979	**1.00**	.974
cardio	.120	*.147*	*.588*	.581	**.588**	.578	.536	*.545*	*.565*	.548
coil1	.100	*.107*	.414	**.426**	.409	*.420*	*.366*	.309	*.384*	.376
covtype	*.839*	.781	**.990**	.989	.984	*.989*	.981	*.985*	.988	*.990*
derma1	.807	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
drybean	*1.00*	*1.00*	**1.00**	**1.00**	1.00	**1.00**	**1.00**	**1.00**	1.00	**1.00**
ecoli1	.726	*.860*	.794	*.808*	*.848*	.818	*.809*	.804	**.927**	.877
ecoli2	.247	*.534*	.820	*.870*	.699	*.832*	.674	**.904**	.794	*.851*
ecoli3	*.837*	.669	.992	**1.00**	*.991*	.990	.994	**1.00**	.996	*.997*
isolet	.548	*.811*	.991	**.995**	*.990*	.988	.981	*.994*	*.990*	.987
kddcup1	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
kddcup2	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
kddcup3	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
kddcup4	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
kddcup5	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
led7digit1	*.590*	.433	**.886**	.853	*.849*	.841	*.814*	.749	*.861*	.836
letter	*.943*	.851	*1.00*	1.00	1.00	*1.00*	*1.00*	.999	**1.00**	1.00
libras	.281	**1.00**	.998	**1.00**	.938	**1.00**	.985	**1.00**	.936	**1.00**
machine	*.694*	.518	**.967**	.949	.952	*.952*	*.941*	.898	*.953*	.950
mgraphy	*.516*	.514	**.944**	.925	*.931*	.919	*.932*	.901	*.931*	.915
oil	*.302*	.188	**.814**	.740	.774	*.793*	*.798*	.734	*.702*	.696
ozone	.153	*.200*	**.826**	.794	*.794*	.754	.651	*.687*	.787	*.796*
page1	**1.00**	.633	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	.980	**1.00**	**1.00**
poker1	.235	*.240*	.922	*.931*	.731	*.909*	.656	**.944**	.806	*.929*
poker2	.259	*.272*	**1.00**	.991	.720	*.991*	.960	*.976*	.441	*.962*
poker3	*.197*	-.008	**1.00**	.992	.914	*.996*	*.994*	.989	.543	*.998*
scene	.122	*.288*	.665	**.709**	.640	*.675*	.526	*.641*	.634	*.693*
shuttle1	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
shuttle2	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
solar1	*.229*	.019	*.415*	.399	.410	*.423*	*.383*	.298	.434	**.464**
spect	.695	*.876*	.933	**.970**	.887	*.957*	.875	*.883*	.944	*.958*
stars	.796	*.854*	**.986**	.971	*.977*	.965	*.973*	.971	*.972*	.966

Table 13 (continued)

Dataset	Decision tree		CatBoost		LightGBM		Random forest		XGBoost	
	X	X_a	X	X_a	X	X_a	X	X_a	X	X_a
thyroid	*.899*	.871	**.998**	.995	*.997*	.994	*.995*	.957	*.996*	.995
uscrime	*.411*	.385	.794	*.799*	.762	*.791*	*.780*	.777	.786	**.803**
vowel	*.906*	.884	**1.00**	**1.00**	**1.00**	**1.00**	.999	**1.00**	1.00	**1.00**
webpage	*.756*	.694	**.963**	.962	.944	*.961*	.940	*.942*	.943	*.953*
wine1	*-.014*	-.018	*.683*	.608	.737	**.871**	*.728*	.682	.703	*.823*
wine2	*-.020*	-.023	**.538**	.398	.433	*.449*	*.469*	.175	.222	*.315*
wine3	-.011	*-.008*	*.636*	.613	.568	**.715**	.460	*.579*	.328	*.595*
wine4	.464	*.465*	.871	**.964**	.852	*.901*	.766	*.959*	.753	*.940*
yeast1	*.623*	.356	*.669*	.642	.593	*.638*	.743	**.754**	.659	*.664*
yeast2	.692	*.725*	*.964*	.963	.873	*.936*	**.976**	.954	*.963*	.957
yeast3	.303	*.438*	**.582**	.564	.503	*.536*	.489	*.554*	.438	*.494*
yeast4	.253	*.626*	**.858**	.814	.770	*.802*	*.775*	.759	*.838*	.798
yeast5	*.336*	.314	*.642*	.525	*.606*	.393	.557	**.655**	*.568*	.512
yeast6	*-.039*	-.046	*.367*	.341	.345	**.418**	*.336*	.080	*.211*	.147
yeast7	*.544*	.266	.797	*.849*	*.753*	.643	.826	**.861**	.741	*.799*
yeast8	*.740*	.558	*.981*	.978	.976	**.986**	*.980*	.959	.970	*.978*
yeast9	.125	*.589*	*.538*	.467	.100	*.490*	**.805**	.425	*.621*	.464
yeast10	*.333*	.194	.838	**.863**	*.857*	.786	*.842*	.782	*.819*	.786
yeast11	*.611*	.589	*.978*	.976	.964	*.973*	**.978**	.977	*.962*	.961
yeast12	.320	*.626*	**.873**	.790	*.842*	.774	.790	*.862*	.691	*.784*
yeast13	.198	*.344*	.843	**.867**	.809	*.846*	.849	*.858*	.841	*.844*
yeast14	-.018	*-.004*	**.237**	.152	.150	*.215*	.038	*.150*	*.151*	.148

Best between X and X_a for each ML model highlighted in italic. Best for each dataset highlighted in bold

Table 14 GM among various datasets and ML models trained on original data X or on data pre-processed with FROID X_a

Dataset	Decision tree		CatBoost		LightGBM		Random forest		XGBoost	
	X	X_a	X	X_a	X	X_a	X	X_a	X	X_a
abalone1	*.228*	.000	*.000*	*.000*	*.244*	.000	*.000*	*.000*	**.244**	.000
abalone2	*.286*	.253	.286	*.576*	**.584**	.441	.000	*.279*	*.286*	.255
abalone3	.341	*.809*	.343	*.602*	.350	*.598*	.342	*.601*	.601	**.814**
abalone4	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
abalone5	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**
abalone6	.222	*.325*	.361	*.431*	**.517**	.510	*.361*	.339	.430	*.512*
arrhythmia	.780	*.934*	.783	**.996**	.685	*.939*	.000	*.792*	.544	**.996**
avila	*.995*	.922	**1.00**	.935	*.976*	.957	**1.00**	.393	**1.00**	.929
bank	.586	**.632**	.544	*.555*	*.615*	.551	*.455*	.427	*.622*	.530
careval1	*.900*	.746	*.924*	.845	**.951**	.907	*.906*	.580	*.901*	.800
careval2	*.999*	.773	**1.00**	.768	**1.00**	.884	*.856*	.764	**1.00**	.801
cardio	.408	**.437**	*.139*	.092	*.121*	.105	*.133*	.113	*.170*	.161
coil1	.378	**.399**	*.130*	.088	*.219*	.176	*.227*	.190	*.264*	.177
covtype	*.917*	.886	**.950**	.936	*.940*	.939	*.922*	.883	*.947*	.932
derma1	.893	**1.00**	.902	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	.902	**1.00**
drybean	*1.00*	*1.00*	1.00	**1.00**	*1.00*	*1.00*	*1.00*	*1.00*	*1.00*	*1.00*
ecoli1	.854	**.927**	.872	*.877*	.734	*.877*	*.802*	*.802*	.816	*.877*
ecoli2	.471	*.719*	*.724*	*.724*	.476	*.581*	.680	*.724*	.473	**.805**
ecoli3	*.911*	.805	**.917**	.809	.804	**.917**	.809	**.917**	.911	**.917**
isolet	.750	*.903*	.906	*.955*	.914	*.949*	.751	*.913*	.899	**.957**
kddcup1	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
kddcup2	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
kddcup3	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
kddcup4	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
kddcup5	**1.00**	**1.00**	**1.00**	**1.00**	.908	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
led7digit1	*.789*	.656	*.789*	.654	*.789*	.735	*.789*	.654	**.792**	.734
letter	*.971*	.922	.964	*.966*	.966	**.984**	*.942*	.883	.966	*.970*
libras	.484	**1.00**	.833	**1.00**	.922	**1.00**	.620	**1.00**	.740	**1.00**
machine	**.835**	.724	*.811*	.701	*.821*	.670	*.713*	.493	*.821*	.727
mgraphy	*.777*	.720	**.799**	.710	*.774*	.735	*.715*	.660	*.788*	.744
oil	*.545*	.449	.544	*.545*	*.459*	.459	**.547**	.546	*.546*	.468
ozone	.403	**.468**	*.000*	*.000*	*.000*	*.000*	*.000*	*.000*	*.000*	*.000*
page1	**1.00**	.790	**1.00**	.860	**1.00**	.996	**1.00**	.699	**1.00**	**1.00**
poker1	**.448**	.442	.000	*.442*	*.280*	.264	.000	*.442*	*.443*	*.443*
poker2	*.474*	.474	*.474*	*.474*	.000	**.863**	.000	*.583*	.000	*.845*
poker3	*.356*	.000	**1.00**	.000	.000	*.751*	.000	*.761*	.000	*.751*
scene	.431	**.583**	.270	*.385*	.270	*.410*	.229	*.270*	.303	*.473*
shuttle1	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
shuttle2	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
solar1	*.361*	.195	*.312*	.000	*.311*	.000	*.000*	*.000*	**.384**	.000
spect	.834	*.937*	.837	**.953**	.882	*.949*	.837	**.953**	.837	*.949*
stars	.894	*.925*	*.918*	.899	*.918*	.902	**.937**	.933	.903	*.919*

Table 14 (continued)

Dataset	Decision tree		CatBoost		LightGBM		Random forest		XGBoost	
	X	X_a	X	X_a	X	X_a	X	X_a	X	X_a
thyroid	*.949*	.934	**.969**	.958	.952	*.957*	*.915*	.692	.945	*.960*
uscrime	**.662**	.647	*.649*	.626	.612	*.643*	*.631*	.576	*.660*	.659
vowel	*.952*	.940	.982	**1.00**	.961	**1.00**	.941	**1.00**	*.982*	.961
webpage	**.862**	.835	*.844*	.841	.789	*.843*	*.839*	.811	.794	*.853*
wine1	*.000*	*.000*	*.000*	*.000*	*.000*	*.000*	*.000*	*.000*	.000	**.781**
wine2	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**
wine3	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**
wine4	.658	*.662*	*.662*	.331	.330	*.537*	*.518*	.509	.536	**.662**
yeast1	**.801**	.635	.693	*.759*	*.761*	.664	.743	*.764*	*.758*	.656
yeast2	.837	*.856*	**.926**	.887	.862	*.888*	*.909*	.868	*.906*	.870
yeast3	.582	**.676**	*.421*	.419	*.550*	.189	*.483*	.419	*.651*	.418
yeast4	.526	**.789**	*.624*	.484	.606	*.704*	.332	*.602*	*.677*	.544
yeast5	**.567**	.561	*.459*	.276	.275	*.430*	*.275*	.238	.279	*.428*
yeast6	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**
yeast7	**.739**	.526	*.541*	*.541*	.417	*.539*	.420	*.541*	.419	*.541*
yeast8	*.864*	.753	.837	*.883*	.917	*.924*	*.883*	.843	.911	**.951**
yeast9	.290	**.749**	*.625*	*.625*	.000	*.625*	*.625*	*.625*	*.625*	*.625*
yeast10	**.581**	.443	*.452*	.351	.423	*.525*	*.528*	.361	*.479*	.350
yeast11	*.779*	.766	**.780**	.777	**.780**	.721	.723	*.778*	**.780**	.670
yeast12	.550	**.787**	*.553*	.000	*.557*	.431	*.553*	.000	*.552*	.479
yeast13	.447	**.595**	*.364*	.000	*.455*	.212	.226	*.321*	*.455*	.339
yeast14	.247	**.255**	*.000*	*.000*	*.000*	*.000*	*.000*	*.000*	*.000*	*.000*

Best between X and X_a for each ML model highlighted in italic. Best for each dataset highlighted in bold

Table 15 F1 Score among various datasets and ML models trained on original data X or on data pre-processed with FROID X_a

Dataset	Decision tree		CatBoost		LightGBM		Random forest		XGBoost	
	X	X_a	X	X_a	X	X_a	X	X_a	X	X_a
abalone1	.061	.000	.000	.000	.154	.000	.000	.000	**.167**	.000
abalone2	.159	.105	.191	.510	**.520**	.361	.000	.203	.209	.188
abalone3	.202	.531	.254	.478	.196	.364	.225	.425	.425	**.677**
abalone4	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
abalone5	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**
abalone6	.105	.127	.245	.299	.355	.364	.245	.243	.283	**.382**
arrhythmia	.752	.877	.757	**.942**	.641	.935	.000	.766	.474	**.942**
avila	.985	.869	**.985**	.917	.961	.946	**.985**	.269	**.985**	.912
bank	.384	.415	.392	.415	.480	.407	.314	.276	**.486**	.381
careval1	.851	.524	.854	.807	**.904**	.869	.845	.472	.840	.757
careval2	.975	.452	**1.00**	.684	**1.00**	.758	.845	.660	**1.00**	.688
cardio	.162	**.185**	.037	.017	.029	.022	.034	.025	.053	.047
coil1	.144	**.156**	.035	.020	.081	.055	.078	.063	.109	.055
covtype	.838	.797	**.914**	.901	.899	.900	.886	.852	.908	.893
derma1	.753	**1.00**	.891	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	.891	**1.00**
drybean	.997	.997	.994	**1.00**	.997	.997	.997	.997	.997	.997
ecoli1	.662	.831	.814	**.866**	.696	**.866**	.778	.778	.796	**.866**
ecoli2	.338	.677	.684	.684	.396	.518	.634	.684	.362	**.782**
ecoli3	.819	.720	**.910**	.786	.708	**.910**	.786	**.910**	.819	**.910**
isolet	.575	.786	.885	.924	.880	.919	.714	.896	.866	**.926**
kddcup1	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
kddcup2	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
kddcup3	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
kddcup4	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
kddcup5	**1.00**	**1.00**	**1.00**	**1.00**	.901	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
led7digit1	.615	.494	.615	.477	.615	.564	.615	.477	**.644**	.558
letter	.950	.883	.963	.966	.965	**.984**	.936	.876	.964	.967
libras	.382	**1.00**	.816	**1.00**	.917	**1.00**	.569	**1.00**	.704	**1.00**
machine	.674	.530	.751	.615	**.760**	.582	.634	.365	.738	.651
mgraphy	.588	.525	**.718**	.619	.680	.643	.635	.579	.706	.647
oil	.324	.197	.375	.394	.326	.310	**.439**	.412	.416	.349
ozone	.180	**.197**	.000	.000	.000	.000	.000	.000	.000	.000
page1	**1.00**	.765	**1.00**	.847	**1.00**	.936	**1.00**	.655	**1.00**	**1.00**
poker1	.252	.277	.000	.335	.204	.179	.000	.294	**.366**	**.366**
poker2	.262	.366	.400	.400	.000	.808	.000	.523	.000	**.830**
poker3	.169	.000	**1.00**	.000	.000	.723	.000	.736	.000	.723
scene	.178	.300	.141	.257	.139	.278	.108	.141	.170	**.342**
shuttle1	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
shuttle2	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
solar1	.175	.062	.161	.000	.154	.000	.000	.000	**.223**	.000
spect	.737	.762	.763	**.890**	.813	.856	.763	**.890**	.763	.856
stars	.973	.978	.984	.976	.984	.981	**.988**	.984	.983	.986

Table 15 (continued)

Dataset	Decision tree		CatBoost		LightGBM		Random forest		XGBoost	
	X	X_a	X	X_a	X	X_a	X	X_a	X	X_a
thyroid	*.862*	.840	**.935**	.874	*.899*	.878	*.887*	.624	.894	*.902*
uscrime	*.431*	.395	*.498*	.477	.454	*.491*	*.495*	.436	.504	**.522**
vowel	.878	*.922*	.982	**1.00**	.960	**1.00**	.939	**1.00**	*.982*	.959
webpage	.657	*.691*	**.794**	.768	.726	*.772*	*.766*	.732	.732	*.774*
wine1	*.000*	*.000*	*.000*	*.000*	*.000*	*.000*	*.000*	*.000*	.000	**.764**
wine2	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**
wine3	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**
wine4	*.462*	.382	*.615*	.267	.244	*.435*	.452	*.454*	.398	**.621**
yeast1	*.589*	.390	.603	*.673*	**.683**	.560	*.675*	.668	*.658*	.530
yeast2	.712	*.717*	*.881*	.838	.788	*.853*	**.889**	.817	*.861*	.832
yeast3	.344	*.489*	*.319*	.302	*.400*	.098	*.375*	.302	**.515**	.287
yeast4	.327	**.712**	*.565*	.386	.486	*.635*	.232	*.510*	*.607*	.438
yeast5	**.380**	.273	*.372*	.202	.170	*.350*	*.183*	.170	.181	*.319*
yeast6	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**
yeast7	**.557**	.297	*.463*	*.463*	.261	*.422*	.301	*.463*	.282	*.463*
yeast8	*.671*	.585	.719	*.814*	.804	**.858**	*.814*	.767	.758	*.817*
yeast9	.164	*.518*	**.575**	**.575**	.000	**.575**	**.575**	**.575**	**.575**	**.575**
yeast10	*.345*	.219	*.329*	.216	.293	*.353*	**.419**	.233	*.329*	.198
yeast11	*.657*	.558	**.687**	.680	**.687**	.650	*.650*	.628	**.687**	.618
yeast12	.369	**.519**	*.476*	.000	*.450*	.327	*.476*	.000	*.445*	.341
yeast13	.214	**.313**	*.199*	.000	*.305*	.099	.118	*.184*	*.304*	.201
yeast14	.065	**.072**	*.000*	*.000*	*.000*	*.000*	*.000*	*.000*	*.000*	*.000*

Best between X and X_a for each ML model highlighted in italic. Best for each dataset highlighted in bold

Table 16 PRA among various datasets obtained by LightGBM trained on different features combinations

Dataset	X	X_a	$X_{\neg\zeta}$	X_σ	X_f	X_l	X_{fo}	X_{fr}	X_O	X_R
abalone1	.115	.098	.089	.096	.092	.161	**.239**	.090	.224	.121
abalone2	**.630**	.521	.535	.538	.545	.529	.440	.529	.484	.446
abalone3	.596	.665	.674	.708	.861	**1.00**	.576	.829	.774	.829
abalone4	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	.837	**1.00**	**1.00**	**1.00**
abalone5	.043	.105	.094	.125	.131	.110	.115	.107	**.145**	.070
abalone6	.380	.515	.547	.461	**.573**	.559	.459	.531	.521	.511
arrhythmia	.924	.979	**1.00**	**1.00**	.896	**1.00**	.891	.896	.896	.896
avila	.992	.983	.966	.966	.584	.986	.335	.498	**.994**	.993
bank	.517	.511	.494	.456	.511	.487	.336	.492	.472	**.519**
careval1	.944	.918	.890	.674	**.978**	**.978**	.366	.961	.926	.962
careval2	**1.00**	.802	.825	.704	.921	.884	.260	.980	.793	.974
cardio	**.203**	.188	.177	.194	.185	.180	.170	.187	.178	.202
coil1	.130	.132	.129	.127	.130	.135	.103	.138	.125	**.139**
covtype	.955	.957	.955	.955	.931	**.957**	.764	.927	.948	.956
derma1	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
drybean1	.999	**1.00**	**1.00**	**1.00**	.000	**1.00**	1.00	**1.00**	**1.00**	.000
ecoli1	.870	**.895**	.882	.872	.867	.863	.683	.822	.834	.822
ecoli2	.528	.778	.778	.778	**.849**	.822	.443	.763	.590	.763
ecoli3	.936	.949	.949	.903	**1.00**	**1.00**	.941	**1.00**	.857	**1.00**
isolet	.961	.969	.973	.973	**.982**	.973	.895	.965	.969	.974
kddcup1	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
kddcup2	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
kddcup3	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
kddcup4	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	.979	**1.00**	**1.00**
kddcup5	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
led7digit1	.615	.725	.723	.696	.783	.621	.566	**.785**	.574	.781
letter	.996	**.996**	.995	.995	.983	.995	.822	.980	.996	.989
libras	.895	**1.00**	**1.00**	**1.00**	.946	.946	**1.00**	.946	.980	.946
machine	**.788**	.734	.727	.705	.572	.771	.327	.542	.780	.718
mgraphy	**.752**	.712	.707	.711	.675	.720	.553	.622	.707	.730
oil	.390	.350	.402	.379	.474	**.530**	.322	.477	.357	.484
ozone	.276	**.313**	.300	.270	.248	.294	.085	.212	.305	.297
page1	**1.00**	**1.00**	**1.00**	**1.00**	.975	**1.00**	.658	.975	**1.00**	.975
poker1	.350	.438	.482	**.576**	.506	.375	.449	.495	.442	.481
poker2	.078	.890	.800	.862	.859	.859	.912	.868	**.929**	.868
poker3	.188	.906	.911	.942	.869	.838	.892	**.951**	.811	**.951**
scene	.331	.431	.403	.409	.486	**.544**	.274	.462	.361	.503
shuttle1	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
shuttle2	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
solar1	**.136**	.128	.107	.110	.124	.120	.109	.104	.113	.117
spect	.865	.898	.907	.892	.944	.946	.768	**.962**	.837	.916
stars	.998	.997	.997	.997	**.998**	.998	.998	.998	.996	.997
thyroid	**.975**	.960	.961	.949	.662	.972	.362	.585	.967	.959
uscrime	.512	.529	.562	.579	.603	.582	.360	**.621**	.522	.566

Table 16 (continued)

Dataset	X	X_a	$X_{\neg\zeta}$	X_σ	X_f	X_l	X_{fo}	X_{fr}	X_O	X_R
vowel	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	.989	**1.00**	.994	**1.00**
webpage	.810	.848	.850	.848	.854	.855	.759	**.871**	.819	.862
wine1	.208	.391	**.428**	.401	.354	.416	.183	.140	.401	.155
wine2	.088	.125	.099	.080	.183	.125	.091	**.249**	.110	.174
wine3	.069	.122	.123	.133	.125	.136	.076	.168	**.174**	.149
wine4	.520	.586	.561	.703	.679	.585	.475	**.712**	.635	.526
yeast1	.611	.601	.604	.563	.582	.617	.662	.569	**.690**	.543
yeast2	.815	.915	.917	**.921**	.897	.898	.839	.878	.889	.909
yeast3	.410	.393	.385	.413	.495	**.535**	.377	.522	.410	.472
yeast4	.592	.733	.730	.709	.720	.716	.480	.740	.600	**.764**
yeast5	.246	.418	**.449**	.370	.296	.310	.105	.239	.315	.416
yeast6	**.188**	.114	.094	.135	.121	.088	.161	.158	.105	.111
yeast7	.403	.491	.485	.485	.575	.459	.266	**.597**	.340	.558
yeast8	.907	.946	**.954**	.894	.884	.938	.680	.878	.875	.937
yeast9	.114	.554	.586	.622	.636	.641	.476	.638	.553	**.642**
yeast10	**.446**	.321	.363	.356	.350	.417	.226	.330	.352	.393
yeast11	.731	.812	**.825**	.810	.770	.800	.588	.814	.782	.813
yeast12	**.663**	.493	.515	.486	.563	.651	.315	.649	.508	.642
yeast13	.301	.240	.258	.228	.246	.266	.255	.303	**.378**	.278
yeast14	.096	.110	.121	.107	.156	.134	.097	**.172**	.103	.149

Best among different features combinations highlighted in bold

Table 17 GINI among various datasets obtained by LightGBM trained on different features combinations

Dataset	X	X_a	$X_{\neg\zeta}$	X_σ	X_f	X_l	X_{fo}	X_{fr}	X_O	X_R
abalone1	.131	.348	.329	.272	.380	.421	.402	.409	.392	**.501**
abalone2	**.968**	.901	.928	.951	.930	.845	.612	.907	.762	.890
abalone3	.967	.963	.966	.973	.978	.955	.973	.967	**.981**	.964
abalone4	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	.993	**1.00**	**1.00**	**1.00**
abalone5	.582	.647	**.688**	.536	.634	.465	.648	.551	.642	.590
abalone6	.642	.735	.711	.633	.649	**.736**	.649	.635	.687	.718
arrhythmia	.986	.997	**1.00**	**1.00**	.997	.984	.806	.997	.888	.997
avila	1.00	1.00	.999	.999	.922	1.00	.845	.897	**1.00**	1.00
bank	**.801**	.779	.785	.762	.684	.756	.574	.675	.769	.755
careval1	**.990**	.985	.983	.915	.905	**.985**	.652	.912	.988	.982
careval2	**1.00**	.987	.986	.975	.973	.992	.707	.974	.986	.990
cardio	.588	.578	.580	**.592**	.558	.571	.504	.575	.576	.581
coil1	.409	.420	.386	.392	.328	**.422**	.215	.294	.389	.369
covtype	.984	**.989**	.988	.988	.982	.987	.935	.980	.983	.986
derma1	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
drybean	1.00	**1.00**	**1.00**	**1.00**	.000	**1.00**	1.00	**1.00**	**1.00**	.000
ecoli1	**.848**	.818	.806	.800	.769	.769	.764	.781	.795	.768
ecoli2	.699	.832	**.833**	**.833**	.815	.787	.737	.710	.759	.739
ecoli3	.991	.990	.990	.964	.994	**.996**	.990	.994	.974	.991
isolet	.990	.988	.988	.988	.989	.988	.965	.980	**.994**	.992
kddcup1	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
kddcup2	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
kddcup3	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
kddcup4	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
kddcup5	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
led7digit1	.849	.841	**.901**	.848	.841	.756	.744	.713	.713	.782
letter	1.00	**1.00**	1.00	1.00	.998	1.00	.951	.996	1.00	1.00
libras	.938	**1.00**	**1.00**	**1.00**	**1.00**	.995	**1.00**	**1.00**	.997	**1.00**
machine	.952	.952	.946	.948	.919	.946	.801	.913	.952	**.960**
mgraphy	.931	.919	.913	.905	.881	.898	.842	.876	.916	**.932**
oil	.774	.793	**.812**	.788	.710	.768	.665	.686	.779	.773
ozone	.794	.754	.775	.810	.546	.787	.371	.604	.779	**.811**
page1	**1.00**	**1.00**	**1.00**	**1.00**	.993	**1.00**	.925	**1.00**	**1.00**	**1.00**
poker1	.731	.909	.880	.916	.894	**.929**	.864	.880	.904	.892
poker2	.720	.991	.972	.979	.977	.980	.995	.830	**.995**	.738
poker3	.914	.996	.997	**.998**	.997	.993	.997	.972	.994	.920
scene	.640	**.675**	.665	.641	.625	.659	.324	.619	.624	.645
shuttle1	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
shuttle2	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
solar1	.410	**.423**	.386	.372	.421	.386	.346	.421	.404	.392
spect	.887	.957	.957	.959	.972	.934	.923	**.979**	.883	.953
stars	**.977**	.965	.964	.964	.970	.956	.970	.965	.954	.970
thyroid	**.997**	.994	.995	.991	.792	.996	.728	.687	.995	.995
uscrime	.762	**.791**	.780	.774	.773	.771	.693	.741	.787	.777

Table 17 (continued)

Dataset	X	X_a	$X_{\neg\zeta}$	X_σ	X_f	X_l	X_{fo}	X_{fr}	X_O	X_R
vowel	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	.999	.997	**1.00**	.999	**1.00**
webpage	.944	.961	.961	.961	.960	.943	.928	**.962**	.948	.960
wine1	.737	.871	.881	.886	**.917**	.383	.809	.836	.889	.695
wine2	.433	.449	.462	.423	.442	.435	**.504**	.449	.473	.418
wine3	.568	.715	.698	.641	.505	**.734**	.370	.717	.637	.708
wine4	.852	.901	.870	.953	.879	.921	.890	.934	.824	**.954**
yeast1	.593	.638	.637	.635	.627	.623	.604	.626	**.683**	.617
yeast2	.873	.936	**.954**	.944	.927	.935	.873	.920	.899	.924
yeast3	.503	.536	.509	.505	.470	**.736**	.319	.479	.400	.508
yeast4	.770	.802	.781	.786	.767	.815	.491	**.879**	.586	.784
yeast5	**.606**	.393	.461	.414	.496	.422	.293	.360	.410	.406
yeast6	.345	.418	.371	.228	.271	.245	**.438**	.067	.181	.113
yeast7	.753	.643	.696	.717	.571	.623	.443	.731	.608	**.770**
yeast8	.976	.986	**.989**	.974	.983	.984	.879	.970	.972	.967
yeast9	.100	.490	.486	**.748**	.580	.280	.340	.635	.367	.627
yeast10	**.857**	.786	.812	.725	.674	.809	.587	.795	.807	.836
yeast11	.964	.973	.971	.975	.977	.975	.886	.980	.968	**.983**
yeast12	.842	.774	.834	.786	.768	.779	.597	.714	**.911**	.718
yeast13	.809	.846	.848	.769	.789	.832	.792	.851	**.859**	.846
yeast14	.150	.215	.245	.244	.201	**.263**	.123	.232	.234	.178

Best among different features combinations highlighted in bold

Table 18 GM among various datasets obtained by LightGBM trained on different features combinations

Dataset	X	X_a	$X_{\neg\zeta}$	X_σ	X_f	X_l	X_{fo}	X_{fr}	X_O	X_R
abalone1	**.244**	.000	.000	.000	.000	.000	.000	.000	.000	.000
abalone2	**.584**	.441	.441	.441	.582	.441	.576	.255	.441	.446
abalone3	.350	.598	.598	.598	.602	.601	.000	.602	**.638**	.600
abalone4	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	.997	**1.00**	**1.00**	**1.00**
abalone5	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**
abalone6	.517	.510	.509	.512	.513	.499	.338	.337	.516	**.589**
arrhythmia	.685	.939	.939	.939	**.996**	.685	.447	.934	.872	.939
avila	.976	.957	.916	.906	.406	**.980**	.352	.450	.966	.956
bank	**.615**	.551	.575	.545	.497	.526	.390	.476	.562	.517
careval1	**.951**	.911	.883	.640	.621	**.897**	.310	.674	.870	.907
careval2	**1.00**	.917	.915	.862	.867	.914	.000	.830	.913	.887
cardio	.121	.105	.089	**.135**	.127	.074	.091	.089	.094	.107
coil1	**.219**	.176	.192	.190	.140	.205	.175	.163	.217	.165
covtype	**.940**	.939	.936	.935	.898	.939	.701	.895	.931	.939
derma1	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	.902	**1.00**	**1.00**	**1.00**	**1.00**
drybean	1.00	1.00	1.00	**1.00**	.000	1.00	.997	1.00	**1.00**	.000
ecoli1	.734	**.877**	**.877**	.872	**.877**	**.877**	.792	.802	.799	**.877**
ecoli2	.476	.581	.724	.724	**.805**	.704	.472	**.805**	.476	**.805**
ecoli3	.804	**.917**	**.917**	.809	**.917**	**.917**	.809	.804	.911	**.917**
isolet	.914	.949	**.949**	.949	.946	.946	.789	.942	.907	.937
kddcup1	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
kddcup2	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
kddcup3	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
kddcup4	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
kddcup5	.908	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	.908	**1.00**	.908
led7digit1	**.789**	.735	.657	.657	.735	.734	.503	.734	.580	.734
letter	.966	**.984**	.979	.976	.965	.982	.700	.965	.959	.981
libras	.922	**1.00**	**1.00**	**1.00**	**1.00**	.833	**1.00**	**1.00**	.922	**1.00**
machine	**.821**	.670	.706	.705	.600	.719	.328	.567	.781	.781
mgraphy	**.774**	.735	.718	.710	.735	.766	.619	.705	.751	.758
oil	.459	.459	.459	.459	.457	.459	.466	.456	.458	**.544**
ozone	.000	.000	.000	.000	.000	**.173**	.000	.000	.000	.000
page1	**1.00**	.996	**1.00**	**1.00**	.780	**1.00**	.575	.920	**1.00**	**1.00**
poker1	.280	.264	.264	**.567**	.264	.264	.280	.428	.442	.252
poker2	.000	.863	.776	.777	**.922**	.777	.862	.746	.922	.749
poker3	.000	**.751**	**.751**	**.751**	**.751**	**.751**	.583	.373	.583	.365
scene	.270	.410	.412	**.512**	.447	.359	.341	.484	.356	.456
shuttle1	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	.999	**1.00**	**1.00**	**1.00**
shuttle2	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	.999	**1.00**	**1.00**
solar1	**.311**	.000	.000	.000	.187	.297	.000	.284	.000	.283
spect	.882	.949	.949	.952	.920	.815	.840	**.956**	.836	.953
stars	.918	.902	.885	.902	.915	.903	.887	.913	.887	**.959**
thyroid	.952	.957	.951	.936	.481	**.981**	.376	.419	.944	.942
uscrime	.612	.643	.664	.626	.643	.664	.328	.664	.594	**.679**

Table 18 (continued)

Dataset	X	X_a	$X_{\neg\zeta}$	X_σ	X_f	X_l	X_{fo}	X_{fr}	X_O	X_R
vowel	.961	**1.00**	**1.00**	**1.00**	.979	.996	.941	.979	.998	.979
webpage	.789	.843	.842	.845	.838	.769	.797	**.846**	.806	.834
wine1	.000	.000	.000	.000	.000	**.446**	.000	.000	.000	.000
wine2	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**
wine3	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**
wine4	.330	.537	.537	**.662**	.331	.537	.331	.538	.538	.537
yeast1	**.761**	.664	.663	.638	.665	.693	.672	.680	.740	.681
yeast2	.862	.888	.888	.888	.888	.888	.852	.884	.870	**.905**
yeast3	.550	.189	.308	.308	.419	.511	.419	.308	**.558**	.308
yeast4	.606	.704	.664	.704	.704	.608	.491	**.756**	.547	.671
yeast5	.275	**.430**	.238	**.430**	.429	.000	.000	.429	.000	.428
yeast6	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**
yeast7	.417	.539	.539	.536	.539	.537	.000	.539	.420	**.541**
yeast8	.917	.924	**.945**	.805	.767	.920	.672	.837	.845	.837
yeast9	.000	**.625**	**.625**	**.625**	**.625**	**.625**	**.625**	**.625**	**.625**	**.625**
yeast10	.423	.525	**.581**	.526	.443	.442	.000	.524	.227	.439
yeast11	.780	.721	.707	.657	.658	.732	.667	.838	.541	**.869**
yeast12	.557	.431	.341	.341	.341	**.564**	.205	.341	.341	.432
yeast13	**.455**	.212	.348	.000	.212	.000	.000	.317	.429	.334
yeast14	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**

Best among different features combinations highlighted in bold

Table 19 F1 score among various datasets obtained by LightGBM trained on different features combinations

Dataset	X	X_a	$X_{\neg\zeta}$	X_σ	X_f	X_l	X_{fo}	X_{fr}	X_O	X_R
abalone1	**.154**	.000	.000	.000	.000	.000	.000	.000	.000	.000
abalone2	**.520**	.361	.361	.361	.420	.361	.510	.188	.361	.369
abalone3	.196	.364	.364	.364	.468	.425	.000	.484	**.606**	.402
abalone4	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	.904	**1.00**	**1.00**	**1.00**
abalone5	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**
abalone6	.355	.364	.346	.384	.404	.338	.225	.217	.351	**.465**
arrhythmia	.641	.935	.935	.935	**.942**	.641	.369	.877	.860	.935
avila	.961	.946	.902	.891	.273	**.964**	.219	.332	.955	.940
bank	**.480**	.407	.432	.393	.348	.363	.231	.319	.415	.356
careval1	**.904**	.831	.808	.529	.506	**.805**	.184	.529	.816	.798
careval2	**1.00**	.829	.788	.709	.761	.807	.000	.704	.807	.799
cardio	.029	.022	.016	**.036**	.031	.012	.017	.016	.018	.023
coil1	**.081**	.055	.065	.061	.040	.070	.053	.050	.075	.052
covtype	.899	.900	.901	.899	.849	.899	.623	.842	.891	**.901**
derma1	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	.891	**1.00**	**1.00**	**1.00**	**1.00**
drybean	.997	.997	.997	**1.00**	.000	.997	.994	.997	**1.00**	.000
ecoli1	.696	**.866**	**.866**	.814	**.866**	**.866**	.675	.778	.728	**.866**
ecoli2	.396	.518	.684	.684	**.782**	.658	.338	**.782**	.396	**.782**
ecoli3	.708	**.910**	**.910**	.786	**.910**	**.910**	.786	.712	.819	**.910**
isolet	.880	.919	**.920**	.919	.919	.908	.757	.907	.872	.904
kddcup1	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
kddcup2	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
kddcup3	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
kddcup4	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
kddcup5	.901	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	.901	**1.00**	.901
led7digit1	**.615**	.564	.505	.505	.564	.558	.334	.558	.412	.558
letter	.965	**.984**	.979	.976	.960	.981	.654	.956	.956	.979
libras	.917	**1.00**	**1.00**	**1.00**	**1.00**	.816	**1.00**	**1.00**	.917	**1.00**
machine	**.760**	.582	.619	.618	.469	.630	.178	.429	.713	.702
mgraphy	.680	.643	.627	.610	.647	**.685**	.486	.597	.664	.661
oil	.326	.310	.326	.312	.286	.313	.298	.264	.297	**.376**
ozone	.000	.000	.000	.000	.000	**.084**	.000	.000	.000	.000
page1	**1.00**	.936	**1.00**	**1.00**	.753	**1.00**	.430	.914	**1.00**	**1.00**
poker1	.204	.179	.195	**.501**	.179	.179	.188	.350	.312	.185
poker2	.000	.808	.712	.752	**.918**	.709	.793	.586	.872	.720
poker3	.000	**.723**	**.723**	**.723**	**.723**	**.723**	.540	.307	.479	.302
scene	.139	.278	.268	**.390**	.315	.217	.208	.335	.225	.326
shuttle1	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	.961	**1.00**	**1.00**	**1.00**
shuttle2	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	.987	**1.00**	**1.00**
solar1	**.154**	.000	.000	.000	.076	.102	.000	.127	.000	.120
spect	.813	.856	.856	.884	.882	.590	.793	**.921**	.758	.890
stars	.984	.981	.979	.981	.981	.982	.981	.977	.981	**.984**
thyroid	.899	.878	.886	.872	.335	**.929**	.234	.256	.876	.871
uscrime	.454	.491	.520	.482	.503	.504	.175	**.551**	.427	.532

Table 19 (continued)

Dataset	X	X_a	$X_{\neg\zeta}$	X_σ	X_f	X_l	X_{fo}	X_{fr}	X_O	X_R
vowel	.960	**1.00**	**1.00**	**1.00**	.979	.959	.938	.979	.977	.979
webpage	.726	.772	.768	**.773**	.763	.693	.701	.767	.725	.759
wine1	.000	.000	.000	.000	.000	**.411**	.000	.000	.000	.000
wine2	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**
wine3	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**
wine4	.244	.435	.435	**.621**	.267	.435	.267	.477	.475	.435
yeast1	**.683**	.560	.549	.528	.568	.584	.601	.576	.663	.542
yeast2	.788	.853	.853	.853	.853	.853	.826	.810	.832	**.858**
yeast3	.400	.098	.192	.192	.302	.399	.302	.192	**.456**	.192
yeast4	.486	.635	.587	.635	.635	.540	.382	**.697**	.430	.594
yeast5	.170	**.350**	.170	**.350**	.323	.000	.000	.320	.000	.277
yeast6	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**
yeast7	.261	.422	.422	.390	.422	.394	.000	.424	.301	**.463**
yeast8	.804	**.858**	.834	.724	.680	.808	.513	.715	.771	.717
yeast9	.000	**.575**	**.575**	**.575**	**.575**	**.575**	**.575**	**.575**	**.575**	**.575**
yeast10	.293	.353	**.398**	.380	.251	.312	.000	.339	.109	.261
yeast11	.687	.650	.664	.568	.603	.661	.472	.722	.408	**.821**
yeast12	**.450**	.327	.233	.233	.237	.440	.125	.237	.233	.328
yeast13	**.305**	.099	.191	.000	.094	.000	.000	.136	.277	.178
yeast14	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**

Best among different features combinations highlighted in bold

Table 20 Average relative importance and ranks by category of features for FROID: X original features, X_o OD over original, X_r FR over original, X_{rr} FR over FR, X_{ro} OD over FR, X_{oo} OD over OD, X_{rr} FR over FR

Dataset	X	X_o	X_r	X_{rr}	X_{ro}	X_{or}	X_{oo}
abalone1	.09	.15	.14	.21	.12	.11	.19
abalone2	.09	.25	.03	.42	.08	.09	.04
abalone3	.15	.32	.09	.38	.01	.02	.03
abalone4	.42	.35	.00	.00	.23	.00	.00
abalone5	.06	.28	.12	.14	.11	.18	.11
abalone6	.11	.32	.05	.27	.06	.15	.05
arrhythmia	.09	.00	.00	.91	.00	.00	.00
avila	.45	.36	.02	.13	.02	.02	.01
bank	.31	.19	.17	.13	.07	.08	.06
careval1	.15	.17	.24	.16	.06	.14	.09
careval2	.39	.04	.33	.17	.03	.03	.00
cardio	.09	.06	.15	.55	.10	.02	.02
coil1	.17	.16	.15	.25	.04	.15	.07
covtype	.58	.06	.21	.13	.02	.00	.00
derma1	.30	.00	.00	.70	.00	.00	.00
drybean	.65	.00	.13	.22	.00	.00	.00
ecoli1	.22	.14	.24	.37	.00	.01	.00
ecoli2	.05	.04	.27	.61	.01	.02	.00
ecoli3	.16	.01	.57	.16	.00	.08	.01
isolet	.23	.04	.11	.30	.31	.01	.00
kddcup1	.02	.98	.00	.00	.00	.00	.00
kddcup2	1.00	.00	.00	.00	.00	.00	.00
kddcup3	.68	.32	.00	.00	.00	.00	.00
kddcup4	.00	1.00	.00	.00	.00	.00	.00
kddcup5	.01	.99	.00	.00	.00	.00	.00
led7digit1	.00	.09	.56	.29	.04	.02	.00
letter	.70	.03	.20	.05	.01	.01	.00
libras	.01	.00	.00	.97	.00	.01	.00
machine	.39	.20	.21	.11	.01	.05	.03
mgraphy	.25	.09	.44	.07	.03	.09	.02
oil	.34	.08	.39	.11	.04	.02	.03
ozone	.75	.11	.02	.04	.00	.06	.02
page1	.27	.02	.36	.35	.00	.00	.00
poker1	.00	.57	.08	.03	.14	.08	.09
poker2	.01	.26	.03	.01	.01	.52	.17
poker3	.01	.64	.01	.07	.01	.24	.02
scene	.61	.01	.09	.21	.05	.01	.02
shuttle1	1.00	.00	.00	.00	.00	.00	.00
shuttle2	1.00	.00	.00	.00	.00	.00	.00
solar1	.00	.20	.30	.26	.08	.13	.01
spect	.30	.19	.20	.21	.09	.01	.01
stars	.08	.09	.03	.68	.07	.02	.02
thyroid	.70	.07	.09	.12	.02	.00	.01
uscrime	.45	.06	.03	.35	.03	.06	.02
vowel	.07	.58	.22	.10	.01	.03	.00
webpage	.06	.05	.13	.68	.03	.03	.02

Table 20 (continued)

Dataset	X	X_o	X_r	X_{rr}	X_{ro}	X_{or}	X_{oo}
wine1	.11	.25	.05	.38	.03	.09	.09
wine2	.24	.24	.11	.18	.05	.13	.05
wineq3	.27	.25	.08	.12	.04	.12	.12
wine4	.15	.33	.09	.21	.07	.09	.05
yeast1	.02	.12	.41	.26	.05	.11	.03
yeast2	.04	.06	.56	.19	.03	.10	.01
yeast3	.04	.23	.33	.20	.05	.08	.07
yeast4	.13	.20	.40	.14	.02	.04	.08
yeast5	.10	.12	.25	.24	.12	.10	.08
yeast6	.11	.17	.21	.17	.01	.12	.21
yeast7	.10	.11	.26	.24	.03	.11	.16
yeast8	.25	.01	.31	.35	.01	.03	.04
yeast9	.02	.03	.51	.08	.24	.07	.05
yeast10	.16	.19	.24	.28	.04	.05	.04
yeast11	.30	.04	.47	.11	.03	.02	.02
yeast12	.04	.08	.53	.18	.03	.12	.03
yeast13	.27	.11	.35	.10	.03	.10	.04
yeast14	.59	.12	.07	.11	.03	.04	.05

Table 21 PRA for various datasets obtained by LightGBM trained after different pre-processing methods

Dataset	ORIG	FROID	XGBOD	ROVE	RUND	CURE	RBO	CCR	ADASYN	SMOTE	SVMSMT	SWIM
abalone1	.115	.098	.078	**.145**	.031	.048	.036	.042	.111	.107	.103	.072
abalone2	**.630**	.521	.564	.508	.014	.345	.330	.416	.327	.359	.343	.366
abalone3	.596	.665	.475	.589	.022	**.842**	.718	.644	.765	.816	.832	.579
abalone4	**1.00**	**1.00**	**1.00**	**1.00**	.032	**1.00**	**1.00**	**1.00**	**1.00**	.837	**1.00**	**1.00**
abalone5	.043	.105	.074	.061	.067	.117	.054	.037	.056	.052	**.233**	.063
abalone6	.380	**.515**	.441	.457	.197	.386	.396	.401	.403	.411	.323	.400
arrhythmia	.924	**.979**	.901	.923	.058	.891	.881	.892	.890	.890	.895	.899
avila	.992	.983	.985	.984	.518	.991	.974	**.998**	.994	.995	.989	.974
bank	.517	.511	.482	.511	.497	.462	.507	.487	.504	.492	**.527**	.522
careval1	.944	.909	.923	**.965**	.820	.932	.941	.959	.959	.960	**.965**	.959
careval2	**1.00**	.820	.949	**1.00**	.638	.967	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
cardio	**.203**	.188	*nan*	.200	.184	.160	.201	.202	.176	.181	.186	.200
coil	.130	.132	.130	.118	.106	.126	.131	.129	.126	.125	.126	**.133**
covtype	.955	**.957**	.951	.947	.889	.941	.931	.949	.923	.948	.946	.934
derma1	**1.00**	**1.00**	**1.00**	.956	.055	.956	.956	**1.00**	.956	.956	**1.00**	.956
drybean	.999	**1.00**	.994	.983	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	.994	.977
ecoli1	.870	**.895**	.840	.874	.138	.765	.849	.852	.830	.776	.849	.878
ecoli2	.528	.778	.641	.647	.084	.668	.789	.695	.696	.745	.799	**.839**
ecoli3	.936	.949	**.973**	.922	.062	**.973**	.950	.948	.937	.946	.946	.939
isolet	.961	**.969**	.951	.958	.884	.934	.959	.955	.945	.955	.961	.956
kddcup1	**1.00**	**1.00**	**1.00**	**1.00**	.665	.665	**1.00**	.955	**1.00**	**1.00**	**1.00**	**1.00**
kddcup2	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	.890	**1.00**	**1.00**	*nan*	**1.00**	**1.00**	**1.00**
kddcup3	**1.00**	**1.00**	**1.00**	**1.00**	.019	**1.00**	**1.00**	**1.00**	*nan*	**1.00**	**1.00**	**1.00**
kddcup4	**1.00**	**1.00**	**1.00**	**1.00**	.012	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
kddcup5	**1.00**	**1.00**	**1.00**	**1.00**	.010	.558	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
led7digit1	.615	**.725**	.520	.582	.230	.496	.569	.505	.564	.595	.634	.571
letter	.996	.996	.995	.992	.985	**.997**	.995	.994	.995	**.997**	.996	.996
libras	.895	**1.00**	.954	.893	.064	.884	.902	.894	.875	.887	.883	.889
machine	.788	.734	.798	.817	.589	.668	.796	**.826**	.800	.780	.821	.793
mgraphy	.752	.712	.766	.743	.553	.267	**.781**	.035	.729	.766	.770	.764
oil	.390	.350	.397	.423	.237	**.490**	.413	.409	.478	.439	.463	.434
ozone	.276	**.313**	.252	.177	.183	.173	.197	.206	.274	.253	.265	.213
page1	**1.00**	**1.00**	**1.00**	**1.00**	.201	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
poker1	.350	**.438**	.409	.229	.013	.116	.302	.224	.094	.078	.207	.302
poker2	.078	.890	.813	.102	.018	.771	.332	**1.00**	.958	.938	.381	**1.00**
poker3	.188	.906	.787	.737	.012	.936	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
scene	.331	**.431**	.355	.313	.240	.322	.342	.353	.329	.325	.341	.337
shuttle1	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
shuttle2	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
solar1	.136	.128	.090	.095	**.173**	.106	.105	.151	.124	.118	.127	.164
spect	.865	.898	.835	.863	.800	.790	.871	.883	.793	.843	**.899**	.753
stars	.998	.997	.996	.997	.996	.997	.998	.997	.998	**.999**	**.999**	.998
tyroid	.975	.960	.966	.974	.897	.944	.980	**.983**	.977	.970	.975	.981
uscrime	.512	.529	.524	.531	.460	.545	.525	.536	.518	.533	.539	**.549**

Table 21 (continued)

Dataset	ORIG	FROID	XGBOD	ROVE	RUND	CURE	RBO	CCR	ADASYN	SMOTE	SVMSMT	SWIM
vowel0	**1.00**	**1.00**	.989	.995	.989	**1.00**	.992	.999	**1.00**	**1.00**	.985	**1.00**
webpage	.810	.848	.811	.831	.587	.727	**.877**	.856	.754	.788	.787	.873
wine1	.208	.391	**.710**	.193	.015	.284	.137	.143	.307	.266	*nan*	.127
wine2	.088	.125	.098	.077	.118	.079	.077	.069	**.137**	.131	.120	.090
wine3	.069	.122	.115	.084	.021	.125	.107	.063	**.149**	.126	.090	.129
wine4	.520	.586	**.771**	.292	.023	.264	.371	.435	.380	.334	.573	.760
yeast1	.611	.601	.681	.632	.610	.618	.697	.663	.557	.583	.654	**.711**
yeast2	.815	.915	**.918**	.865	.696	.776	.853	.830	.828	.883	.866	.833
yeast3	.410	.393	.465	.522	.435	**.564**	.523	.545	.511	.504	.481	.522
yeast4	.592	**.733**	.632	.664	.704	.664	.730	.690	.635	.674	.675	.723
yeast5	.246	.418	**.448**	.250	.131	.211	.323	.423	.298	.236	.335	.249
yeast6	.188	.114	.083	**.211**	.092	.086	.149	.206	.148	.128	.116	.147
yeast7	.403	.491	.437	.467	.190	.304	.419	.424	.383	.395	**.534**	.429
yeast8	.907	**.946**	.892	.875	.850	.845	.891	.890	.889	.849	.907	.915
yeast9	.114	.554	.476	.532	.042	.519	.544	.517	.537	.565	**.663**	.553
yeast10	.446	.321	.408	.509	.187	.496	.485	**.550**	.500	.518	.405	.411
yeast11	.731	.812	.833	.771	.601	.786	.774	.723	.817	.800	.804	**.905**
yeast12	.663	.493	.414	.664	.246	.635	**.727**	.610	.570	.549	.652	.575
yeast13	.301	.240	.328	.258	.196	**.395**	.320	.307	.240	.291	.370	.317
yeast14	.096	.110	.091	.101	.141	.129	.110	**.146**	.102	.100	.098	.110

The best approach for each dataset is highlighted in bold

Table 22 GINI for various datasets obtained by LightGBM trained after different pre-processing methods

Dataset	ORIG	FROID	XGBOD	ROVE	RUND	CURE	RBO	CCR	ADASYN	SMOTE	SVMSMT	SWIM
abalone1	.131	.348	**.518**	.194	.118	.192	.172	.273	.232	.314	.239	.211
abalone2	**.968**	.901	.869	.627	.000	.841	.796	.874	.768	.772	.913	.834
abalone3	.967	.963	.958	.891	.000	**.983**	.982	.957	.926	.974	.980	.969
abalone4	**1.00**	**1.00**	**1.00**	**1.00**	.000	**1.00**	**1.00**	**1.00**	**1.00**	.993	**1.00**	**1.00**
abalone5	.582	.647	.575	.422	.591	.701	.623	.584	**.737**	.710	.640	.588
abalone6	.642	**.735**	.643	.709	.632	.582	.486	.592	.545	.600	.465	.698
arrhythmia	.986	**.997**	.921	.975	.000	.807	.876	.943	.788	.782	.878	.907
avila	**1.00**	**1.00**	**1.00**	**1.00**	.992	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	.999
bank	.801	.779	.782	.803	.770	.782	.803	.807	.786	.785	.801	**.811**
careval1	.990	.985	.987	**.994**	.973	.989	.992	.993	.993	.992	**.994**	**.994**
careval2	**1.00**	.987	.995	**1.00**	.971	.997	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
cardio	.588	.578	*nan*	.578	.571	.541	**.602**	.600	.557	.564	.575	.595
coil	.409	.420	.383	.372	.354	.420	**.431**	.411	.393	.403	.417	.409
covtype	.984	**.989**	.987	.984	.979	.981	.980	.987	.983	.983	.986	.981
derma1	**1.00**	**1.00**	**1.00**	.993	.000	.993	.993	**1.00**	.993	.993	**1.00**	.993
drybean	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
ecoli1	.848	.818	.826	.840	.314	.780	.838	.859	**.873**	.794	.864	.862
ecoli2	.699	.832	.817	.662	.000	.833	.835	.835	.637	.732	.839	**.927**
ecoli3	.991	.990	**.996**	.988	.000	**.996**	.990	.990	.986	.989	.989	.987
isolet	.990	.988	.990	.991	.971	.986	.990	.990	.986	.991	**.992**	.989
kddcup1	**1.00**	**1.00**	**1.00**	**1.00**	.993	.993	**1.00**	.999	**1.00**	**1.00**	**1.00**	**1.00**
kddcup2	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	.996	**1.00**	**1.00**	*nan*	**1.00**	**1.00**	**1.00**
kddcup3	**1.00**	**1.00**	**1.00**	**1.00**	.000	**1.00**	**1.00**	**1.00**	*nan*	**1.00**	**1.00**	**1.00**
kddcup4	**1.00**	**1.00**	**1.00**	**1.00**	.000	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
kddcup5	**1.00**	**1.00**	**1.00**	**1.00**	.000	.992	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
led7digit1	.849	.841	.625	.831	.621	.851	.838	**.863**	.823	.855	.850	.841
letter	**1.00**	**1.00**	**1.00**	.999	.998	**1.00**	.999	.999	**1.00**	**1.00**	**1.00**	**1.00**
libras	.938	**1.00**	.992	.933	.000	.890	.953	.936	.772	.907	.886	.917
machine	.952	.952	.954	.962	.937	.928	.954	.962	.950	.943	.958	**.965**
mgraphy	.931	.919	.913	.898	.915	.520	.945	.355	.922	**.947**	.942	.909
oil	.774	.793	.735	.662	.698	**.823**	.710	.785	.751	.743	.806	.804
ozone	.794	.754	.802	.726	.675	.786	.724	.708	**.825**	.801	.812	.789
page1	**1.00**	**1.00**	**1.00**	**1.00**	.690	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
poker1	.731	**.909**	.863	.425	.000	.450	.662	.589	.727	.779	.863	.739
poker2	.720	.991	.985	.197	.000	.963	.847	**1.00**	.998	.996	.916	**1.00**
poker3	.914	.996	.993	.986	.000	.998	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
scene	.640	**.675**	.637	.595	.636	.587	.625	.662	.626	.645	.658	.665
shuttle1	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
shuttle2	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
solar1	.410	**.423**	.324	.310	.410	.404	.361	.420	.374	.340	.397	**.423**
spect	.887	.957	.910	.861	.917	.823	.851	.946	.832	**.964**	.956	.899
stars	.977	.965	.950	.962	.958	.962	.974	.969	.980	.981	**.984**	.973
tyroid	.997	.994	.996	.996	.986	.992	.997	**.998**	.997	.996	.997	.997
uscrime	.762	.791	.778	.801	.750	.789	.763	.797	.802	.794	.782	**.813**

Table 22 (continued)

Dataset	ORIG	FROID	XGBOD	ROVE	RUND	CURE	RBO	CCR	ADASYN	SMOTE	SVMSMT	SWIM
vowel0	**1.00**	**1.00**	.998	.999	.997	**1.00**	.998	**1.00**	**1.00**	**1.00**	.995	**1.00**
webpage	.944	.961	.963	.966	.927	.960	**.973**	.968	.950	.956	.951	.970
wine1	.737	**.871**	.859	.494	.000	.703	.402	.659	.761	.780	*nan*	.620
wine2	.433	.449	.447	.321	.286	.346	.402	.293	.452	**.489**	.375	.453
wine3	.568	.715	.533	.617	.000	.698	**.748**	.454	.742	.607	.584	.745
wine4	.852	.901	**.956**	.815	.000	.529	.804	.913	.721	.782	.891	.887
yeast1	.593	.638	.679	.704	.540	.714	**.744**	.714	.638	.653	.708	.735
yeast2	.873	.936	.924	.949	.837	.940	**.960**	.950	.946	.953	.954	.948
yeast3	.503	.536	.479	.481	**.692**	.576	.502	.508	.467	.522	.521	.510
yeast4	.770	.802	.694	.795	**.879**	.831	.807	.770	.851	.832	.758	.842
yeast5	.606	.393	.461	**.675**	.394	.511	.596	.655	.646	.570	.595	.494
yeast6	.345	**.418**	.215	.346	.408	.192	.327	.369	.345	.319	.317	.328
yeast7	.753	.643	.697	.765	.278	.459	.782	.766	.606	.584	**.808**	.715
yeast8	.976	**.986**	.977	.972	.964	.958	.975	.973	.974	.967	.978	.980
yeast9	.100	.490	.402	.581	.000	.580	.625	.546	.663	.692	**.810**	.635
yeast10	.857	.786	**.875**	.724	.765	.840	.854	.837	.835	.785	.839	.852
yeast11	.964	.973	.956	.844	.958	.969	.840	.966	.975	.974	.974	**.991**
yeast12	.842	.774	.696	.725	.749	.882	.828	.850	.829	.841	.648	**.898**
yeast13	.809	.846	**.860**	.760	.768	.752	.812	.800	.773	.752	.796	.827
yeast14	.150	.215	.188	.187	.284	.207	.278	.275	.174	.152	.218	**.309**

The best approach for each dataset is highlighted in bold

Table 23 GM for various datasets obtained by LightGBM trained after different pre-processing methods

Dataset	ORIG	FROID	XGBOD	ROVE	RUND	CURE	RBO	CCR	ADASYN	SMOTE	SVMSMT	SWIM
abalone1	**.244**	.000	.000	.000	.000	.000	.000	.000	**.244**	.000	.243	.000
abalone2	.584	.441	.441	.584	.000	**.591**	.254	.255	.439	.254	.254	.255
abalone3	.350	.598	.601	.604	.000	**.809**	.344	.602	.807	.807	.807	.602
abalone4	**1.00**	**1.00**	**1.00**	**1.00**	.000	**1.00**	**1.00**	.997	**1.00**	.997	**1.00**	**1.00**
abalone5	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**
abalone6	.517	.510	.500	.433	.000	.440	.431	.224	.635	**.670**	.429	.224
arrhythmia	.685	**.939**	.706	.544	.000	.938	.691	.427	.706	.796	.427	.691
avila	.976	.957	.950	.971	**.988**	.966	.976	.967	.966	.980	.961	.000
bank	.615	.551	.553	.407	**.734**	.464	.224	.237	.325	.351	.427	.235
careval1	.951	.911	.828	.951	.832	**.990**	.915	.887	.927	.939	.885	.903
careval2	**1.00**	.917	.915	**1.00**	.000	.942	.973	.973	.971	**1.00**	.915	.973
cardio	.121	.105	*nan*	.000	**.304**	.000	.055	.000	.000	.000	.035	.000
coil	.219	.176	.217	.091	**.382**	.085	.000	.000	.000	.000	.000	.000
covtype	.940	.939	.930	.904	.939	.891	**.941**	.743	.825	.884	.891	.681
derma1	**1.00**	**1.00**	.902	.902	.000	.902	.902	**1.00**	.902	.902	.984	.902
drybean	**1.00**	**1.00**	**1.00**	**1.00**	.999	**1.00**	**1.00**	.997	**1.00**	**1.00**	**1.00**	**1.00**
ecoli1	.734	**.877**	.816	.872	.000	.793	.872	.738	.798	.798	.872	.648
ecoli2	.476	.581	.473	.474	.000	.597	**.805**	.474	.600	.597	.600	.724
ecoli3	.804	**.917**	**.917**	.809	.000	.911	.809	**.917**	.809	**.917**	**.917**	.809
isolet	.914	**.949**	.912	.870	.948	.929	.919	.849	.856	.851	.898	.818
kddcup1	**1.00**	**1.00**	**1.00**	**1.00**	.997	.997	**1.00**	.865	**1.00**	**1.00**	**1.00**	**1.00**
kddcup2	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	.998	**1.00**	**1.00**	*nan*	**1.00**	**1.00**	**1.00**
kddcup3	**1.00**	**1.00**	**1.00**	**1.00**	.000	**1.00**	**1.00**	**1.00**	*nan*	**1.00**	**1.00**	**1.00**
kddcup4	**1.00**	**1.00**	**1.00**	**1.00**	.000	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
kddcup5	.908	**1.00**	**1.00**	**1.00**	.000	.996	**1.00**	**1.00**	.999	**1.00**	**1.00**	**1.00**
led7digit1	**.789**	.735	.734	.734	.000	**.789**	.416	.786	.414	.737	**.789**	.414
letter	.966	**.984**	.964	.959	.963	.943	.940	.906	.932	.940	.950	.905
libras	.922	**1.00**	.739	.739	.000	.919	.833	.833	.837	.918	.922	.837
machine	.821	.670	.792	.766	**.828**	.532	.527	.564	.731	.722	.767	.527
mgraphy	.774	.735	.759	.695	**.884**	.600	.687	.426	.630	.658	.715	.596
oil	.459	.459	.466	.459	.539	**.691**	.459	.459	.557	.460	.459	.459
ozone	.000	.000	.275	.000	**.406**	.162	.000	.000	.169	.163	.000	.149
page1	**1.00**	.996	**1.00**	**1.00**	.000	**1.00**	**1.00**	**1.00**	.996	.996	**1.00**	.996
poker1	.280	.264	**.428**	.000	.000	.000	.000	.000	.000	.000	.000	.000
poker2	.000	.863	.699	.000	.000	.281	.000	.250	.821	**.908**	.000	.611
poker3	.000	.751	.583	.891	.000	.744	.583	.340	.875	.875	.340	**1.00**
scene	.270	.410	.383	.270	**.576**	.482	.270	.270	.338	.302	.305	.229
shuttle1	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
shuttle2	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
solar1	.311	.000	.000	.000	.000	**.421**	.000	.000	.000	.000	.000	.000
spect	.882	**.949**	.834	.830	.745	.856	.784	.784	.832	.878	.875	.781
stars	.918	.902	.851	.932	.912	.906	.899	.917	.932	.922	**.962**	**.962**
tyroid	.952	.957	.945	.960	**.963**	.942	.947	.940	.953	.960	.946	.893
uscrime	.612	.643	.625	.617	**.783**	.685	.595	.614	.612	.627	.629	.534

Table 23 (continued)

Dataset	ORIG	FROID	XGBOD	ROVE	RUND	CURE	RBO	CCR	ADASYN	SMOTE	SVMSMT	SWIM
vowel0	.961	**1.00**	.976	.980	.975	.998	.980	.982	.982	.999	.976	**1.00**
webpage	.789	**.843**	.767	.723	.757	.663	.835	.563	.445	.521	.569	.535
wine1	.000	.000	**.488**	.000	.000	.000	.000	.000	.000	.000	*nan*	.000
wine2	.000	.000	.000	.000	.195	**.318**	.000	.000	.000	.000	.197	.000
wine3	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**
wine4	.330	.537	**.789**	.000	.000	.319	.331	.515	.000	.330	.331	.662
yeast1	.761	.664	.759	.722	.746	.714	.642	.589	.692	.691	**.784**	.615
yeast2	.862	.888	**.926**	.906	.894	.896	.799	.865	.859	.882	.886	.906
yeast3	.550	.189	.556	.506	.660	**.667**	.421	.508	.421	.418	.418	.419
yeast4	.606	**.704**	.547	.510	.664	.667	.510	.437	.569	.560	.560	.625
yeast5	.275	.430	**.459**	.457	.000	.456	.279	.000	.279	.000	**.459**	.279
yeast6	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**
yeast7	.417	.539	.539	.532	.000	.392	.419	.419	.415	.415	**.545**	.417
yeast8	.917	**.924**	.921	.917	.764	.867	.840	.886	.880	.921	.914	.840
yeast9	.000	**.625**	**.625**	.293	.000	**.625**	**.625**	**.625**	.000	.293	**.625**	**.625**
yeast10	.423	.525	.577	.418	.595	**.629**	.350	.205	.520	.524	.515	.205
yeast11	.780	.721	.868	.780	.442	**.908**	.780	.735	.868	.828	.725	.868
yeast12	.557	.431	.341	.658	.205	**.765**	.473	.473	.346	.528	.552	.197
yeast13	.455	.212	.363	.365	.204	**.529**	.226	.000	.365	.454	.455	.226
yeast14	.000	.000	.000	.000	**.266**	.114	.000	.000	.000	.000	.000	.000

The best approach for each dataset is highlighted in bold

Table 24 F1 for various datasets obtained by LightGBM trained after different pre-processing methods

Dataset	ORIG	FROID	XGBOD	ROVE	RUND	CURE	RBO	CCR	ADASYN	SMOTE	SVMSMT	SWIM
abalone1	**.154**	.000	.000	.000	.000	.000	.000	.000	.144	.000	.125	.000
abalone2	**.520**	.361	.361	**.520**	.000	.395	.164	.188	.281	.163	.149	.188
abalone3	.196	.364	.425	**.564**	.000	.536	.307	.478	.491	.493	.491	.478
abalone4	**1.00**	**1.00**	**1.00**	**1.00**	.000	**1.00**	**1.00**	.904	**1.00**	.904	**1.00**	**1.00**
abalone5	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**
abalone6	.355	.364	.354	.317	.000	.249	.300	.138	.438	**.496**	.269	.128
arrhythmia	.641	**.935**	.661	.474	.000	.934	.645	.346	.661	.771	.346	.645
avila	.961	.946	.934	.960	.457	.937	**.965**	.961	.946	**.965**	.950	.000
bank	.480	.407	.403	.266	**.521**	.315	.094	.104	.176	.203	.282	.103
careval1	**.904**	.831	.780	**.904**	.756	.895	.878	.847	.891	.891	.845	.864
careval2	**1.00**	.829	.847	**1.00**	.000	.895	.972	.972	.971	**1.00**	.911	.972
cardio	.029	.022	*nan*	.000	**.133**	.000	.007	.000	.000	.000	.004	.000
coil	.081	.055	.074	.021	**.142**	.019	.000	.000	.000	.000	.000	.000
covtype	.899	**.900**	.891	.873	.844	.856	.857	.708	.791	.858	.864	.630
derma1	**1.00**	**1.00**	.891	.891	.000	.891	.891	**1.00**	.891	.891	.775	.891
drybean	.997	.997	.997	.997	.987	**1.00**	.997	.991	.997	**1.00**	.997	.997
ecoli1	.696	**.866**	.796	.814	.000	.681	.814	.704	.729	.729	.814	.592
ecoli2	.396	.518	.362	.366	.000	.490	**.782**	.366	.536	.496	.536	.684
ecoli3	.708	**.910**	**.910**	.786	.000	.819	.786	**.910**	.786	**.910**	**.910**	.786
isolet	.880	**.919**	.871	.843	.783	.874	.885	.821	.827	.826	.877	.794
kddcup1	**1.00**	**1.00**	**1.00**	**1.00**	.791	.791	**1.00**	.796	**1.00**	**1.00**	**1.00**	**1.00**
kddcup2	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	.940	**1.00**	**1.00**	*nan*	**1.00**	**1.00**	**1.00**
kddcup3	**1.00**	**1.00**	**1.00**	**1.00**	.000	**1.00**	**1.00**	**1.00**	*nan*	**1.00**	**1.00**	**1.00**
kddcup4	**1.00**	**1.00**	**1.00**	**1.00**	.000	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
kddcup5	.901	**1.00**	**1.00**	**1.00**	.000	.704	**1.00**	**1.00**	.928	**1.00**	**1.00**	**1.00**
led7digit1	.615	.564	.558	.558	.000	**.619**	.321	.592	.300	.585	.615	.300
letter	.965	**.984**	.959	.958	.933	.941	.938	.901	.930	.938	.948	.901
libras	.917	**1.00**	.703	.703	.000	.861	.816	.816	.820	.857	.917	.820
machine	**.760**	.582	.722	.718	.554	.419	.432	.483	.693	.669	.710	.429
mgraphy	**.680**	.643	.672	.621	.546	.071	.610	.037	.535	.574	.639	.520
oil	.326	.310	.311	.326	.306	**.529**	.326	.326	.430	.331	.326	.326
ozone	.000	.000	.154	.000	**.162**	.053	.000	.000	.079	.073	.000	.073
page1	**1.00**	.936	**1.00**	**1.00**	.000	**1.00**	**1.00**	**1.00**	.940	.940	**1.00**	.936
poker1	.204	.179	**.324**	.000	.000	.000	.000	.000	.000	.000	.000	.000
poker2	.000	.808	.659	.000	.000	.209	.000	.177	.804	**.901**	.000	.551
poker3	.000	.723	.540	.711	.000	.719	.526	.277	.865	.865	.277	**1.00**
scene	.139	.278	.248	.141	.267	**.305**	.141	.141	.197	.163	.171	.108
shuttle1	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
shuttle2	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
solar1	.154	.000	.000	.000	.000	**.168**	.000	.000	.000	.000	.000	.000
spect	.813	**.856**	.738	.755	.660	.679	.702	.702	.757	.780	.806	.678
stars	.984	.981	.976	.984	.961	.956	.979	.968	.969	.974	**.986**	.975
tyroid	.899	.878	.891	.902	.791	.868	**.914**	.913	**.914**	.913	.908	.874
uscrime	.454	.491	.454	**.495**	.493	.483	.456	.468	.472	.490	.469	.425

Table 24 (continued)

Dataset	ORIG	FROID	XGBOD	ROVE	RUND	CURE	RBO	CCR	ADASYN	SMOTE	SVMSMT	SWIM
vowel0	.960	**1.00**	.924	.967	.919	.984	.967	.982	.982	.986	.927	**1.00**
webpage	.726	.772	.699	.670	.615	.576	**.785**	.482	.328	.424	.485	.446
wine1	.000	.000	**.455**	.000	.000	.000	.000	.000	.000	.000	*nan*	.000
wine2	.000	.000	.000	.000	.079	**.129**	.000	.000	.000	.000	.111	.000
wine3	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**
wine4	.244	.435	**.764**	.000	.000	.172	.267	.338	.000	.222	.267	.621
yeast1	.683	.560	.682	.636	.591	.503	.552	.496	.580	.589	**.710**	.528
yeast2	.788	.853	**.881**	.861	.778	.780	.728	.802	.749	.797	.790	.860
yeast3	.400	.098	.435	.399	.496	**.550**	.319	.419	.319	.287	.287	.303
yeast4	.486	**.635**	.428	.424	.586	.524	.421	.337	.423	.464	.464	.541
yeast5	.170	.350	**.372**	.316	.000	.279	.183	.000	.179	.000	**.372**	.167
yeast6	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**
yeast7	.261	.422	.422	.344	.000	.185	.280	.280	.241	.241	**.427**	.259
yeast8	.804	**.858**	.833	.804	.686	.692	.743	.793	.744	.810	.781	.743
yeast9	.000	**.575**	**.575**	.230	.000	**.575**	**.575**	**.575**	.000	.230	**.575**	**.575**
yeast10	.293	.353	**.409**	.313	.241	.381	.239	.116	.359	.362	.374	.116
yeast11	.687	.650	**.786**	.687	.329	.743	.687	.666	**.786**	.742	.623	**.786**
yeast12	.450	.327	.233	**.602**	.111	**.602**	.385	.385	.231	.423	.446	.128
yeast13	.305	.099	.180	.219	.102	**.349**	.118	.000	.219	.279	.305	.134
yeast14	.000	.000	.000	.000	**.098**	.037	.000	.000	.000	.000	.000	.000

The best approach for each dataset is highlighted in bold

Acknowledgements This work is partially supported by SoBigData.it that receives funding from European Union - NextGenerationEU - National Recovery and Resilience Plan (Piano Nazionale di Ripresa e Resilienza, PNRR) - Project: "SoBigData.it - Strengthening the Italian RI for Social Mining and Big Data Analytics" - Prot. IR0000013 - Avviso n. 3264 del 28/12/2021, and by PNRR - M4C2 - Investimento 1.3, Partenariato Esteso PE00000013 - "FAIR - Future Artificial Intelligence Research" - Spoke 1 "Human-centered AI", funded by the European Commission under the NextGeneration EU programme, and by the funding schemes PNRR-PE-AI FAIR (Future Artificial Intelligence Research), Horizon 2020 Program under the scheme "INFRAIA-01-2018-2019 - Integrating Activities for Advanced Communities", Grant Agreement n.871042, "SoBigData++: European Integrated Infrastructure for Social Mining and Big Data Analytics" (http://www.sobigdata.eu).

Author contribution SL—Conceptualization, Methodology, Software, Validation, Investigation, Resources, Writing—Original Draft. AP—Methodology, Software, Validation, Investigation, Resources, Writing—Review & Editing, Visualization. RG—Conceptualization, Methodology, Validation, Investigation, Resources, Data Curation, Writing— Original Draft, Writing—Review & Editing, Visualization.

Funding Open access funding provided by Università di Pisa within the CRUI-CARE Agreement.

Data availability The open-source datasets adopted in this work are available at https://archive.ics.uci.edu/, https://www.kaggle.com/datasets, http://odds.cs.stonybrook.edu/, https://generali.datachallenge.it/, https://sci2s.ugr.es/keel/datasets.php, https://imbalanced-learn.org/stable/datasets/index.htm.

Code availability The code is open source, and can be downloaded at https://github.com/andrepugni/FROID.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

Ethics approval Not applicable.

Consent to participate Not applicable.

Consent for publication The authors declare that they all provide consent for publication.

References

Achtert, E., Böhm, C., Kröger, P., Kunath, P., Pryakhin, A., Renz, M. (2006). Efficient reverse k-nearest neighbor search in arbitrary metric spaces. In SIGMOD conference (pp. 515–526). ACM.

Akbani, R., Kwek, S., & Japkowicz, N. (2004). Applying support vector machines to imbalanced datasets. *ECML. Lecture notes in computer science* (Vol. 3201, pp. 39–50). Springer.

Bellinger, C., Branco, P., & Torgo, L. (2019). The CURE for class imbalance. *DS. Lecture notes in computer science* (Vol. 11828, pp. 3–17). Springer.

Bellinger, C., Corizzo, R., & Japkowicz, N. (2021). Calibrated resampling for imbalanced and long-tails in deep learning. *DS. Lecture notes in computer science* (Vol. 12986, pp. 242–252). Springer.

Bengio, Y., Delalleau, O., Roux, N. L., Paiement, J., Vincent, P., & Ouimet, M. (2006). Spectral dimensionality reduction. *Feature extraction. Studies in fuzziness and soft computing* (Vol. 207, pp. 519–550). Springer.

Branco, P., Torgo, L., & Ribeiro, R. P. (2016). A survey of predictive modeling on imbalanced domains. *ACM Computing Surveys, 49*(2), 31–13150.

Branco, P., Torgo, L., & Ribeiro, R. P. (2018). Resampling with neighbourhood bias on imbalanced domains. *Expert Systems: The Journal of Knowledge Engineering, 35*(4), e12311.

Breiman, L., Friedman, J. H., Olshen, R. A., Stone, C. J. (1984). Classification and regression trees. Wadsworth.

Breiman, L. (2001). Random forests. *Machine Learning, 45*(1), 5–32.

Breunig, M. M., Kriegel, H., Ng, R. T., Sander, J. (2000). LOF: Identifying density-based local outliers. In SIGMOD Conference (pp. 93–104). ACM.

Cano, J. R. (2013). Analysis of data complexity measures for classification. *Expert Systems with Applications, 40*(12), 4820–4831.

Chandola, V., Banerjee, A., & Kumar, V. (2009). Anomaly detection: A survey. *ACM Computing Surveys, 41*(3), 15–11558.

Chawla, N. V. (2010). Data mining for imbalanced datasets: An overview. *Data mining and knowledge discovery handbook* (pp. 875–886). Springer.

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

Chawla, N. V., Japkowicz, N., & Kotcz, A. (2004). Editorial: Special issue on learning from imbalanced data sets. *ACM SIGKDD Explorations Newsletter, 6*(1), 1–6.

Chen, T., Guestrin, C. (2016). Xgboost: A scalable tree boosting system. In KDD (pp. 785–794). ACM.

Cox, M. A., & Cox, T. F. (2008). Multidimensional scaling. *Handbook of data visualization* (pp. 315–347). Springer.

Demsar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine Learning Research, 7*, 1–30.

Donoho, D. L., & Grimes, C. (2003). Hessian eigenmaps: Locally linear embedding techniques for high-dimensional data. *Proceedings of the National Academy of Sciences, 100*(10), 5591–5596.

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

Esenogho, E., Ibomoiye, D. M., Swart, T. G., Aruleba, K. D., & Obaido, G. (2022). A neural network ensemble with feature engineering for improved credit card fraud detection. *IEEE Access, 10*, 16400–16407.

Ester, M., Kriegel, H., Sander, J., Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In KDD (pp. 226–231). AAAI Press.

Fernández, Á., Bella, J., & Dorronsoro, J. R. (2022). Supervised outlier detection for classification and regression. *Neurocomputing, 486*, 77–92.

Goldstein, M., & Dengel, A. (2012). Histogram-based outlier score (hbos): A fast unsupervised anomaly detection algorithm. *KI-2012: Poster and Demo Track, 9*, 59.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. C., Bengio, Y. (2014). Generative adversarial networks. CoRR **abs/1406.2661**

Gopi, S. C., Suvarna, B., & Padmaja, T. M. (2016). High dimensional unbalanced data classification vs svm feature selection. *Indian Journal of Science and Technology, 9*, 30.

Guidotti, R., Monreale, A., Ruggieri, S., Turini, F., Giannotti, F., & Pedreschi, D. (2018). A survey of methods for explaining black box models. *ACM Computing Surveys (CSUR), 51*(5), 1–42.

Guo, H., & Viktor, H. L. (2004). Learning from imbalanced data sets with boosting and data generation: The databoost-im approach. *SIGKDD Explorations Newsletter, 6*(1), 30–39.

Hart, P. E. (1968). The condensed nearest neighbor rule (corresp.). *IEEE Transactions on Information Theory, 14*(3), 515–516.

Hasan, B. M. S., & Abdulazeez, A. M. (2021). A review of principal component analysis algorithm for dimensionality reduction. *Journal of Soft Computing and Data Mining, 2*(1), 20–30.

Hassanat, A. B., Tarawneh, A. S., Altarawneh, G. A. (2022). Stop oversampling for class imbalance learning: A critical review. CoRR **abs/2202.03579**

He, H., Bai, Y., Garcia, E. A., Li, S.(2008). ADASYN: Adaptive synthetic sampling approach for imbalanced learning. In IJCNN (pp. 1322–1328). IEEE.

He, H., & Garcia, E. A. (2009). Learning from imbalanced data. *IEEE Transactions on Knowledge and Data, 21*(9), 1263–1284.

He, Z., Xu, X., & Deng, S. (2003). Discovering cluster-based local outliers. *Pattern Recognition Letters, 24*(9–10), 1641–1650.

He, X., Zhao, K., & Chu, X. (2021). AutoML: A survey of the state-of-the-art. *Knowledge-Based Systems, 212*, 106622.

Hodge, V. J., & Austin, J. (2004). A survey of outlier detection methodologies. *Artificial Intelligence Review, 22*(2), 85–126.

Hubert, M., & Debruyne, M. (2010). Minimum covariance determinant. *Wiley Interdisciplinary Reviews: Computational Statistics, 2*(1), 36–43.

Hubert, M., Debruyne, M., & Rousseeuw, P. J. (2018). Minimum covariance determinant and extensions. *Wiley Interdisciplinary Reviews: Computational Statistics, 10*(3), 1421.

Ibrahim, M. H. (2021). ODBOT: Outlier detection-based oversampling technique for imbalanced datasets learning. *Neural Computing and Applications, 33*(22), 15781–15806.

Japkowicz, N., & Stephen, S. (2002). The class imbalance problem: A systematic study. *Intelligent Data Analysis, 6*(5), 429–449.

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., Liu, T.(2017). Lightgbm: A highly efficient gradient boosting decision tree. In NIPS (pp. 3146–3154).

Khan, K., Rehman, S. U., Aziz, K., Fong, S., Sarasvady, S. (2014). Dbscan: Past, present and future. In The 5th international conference on the applications of digital information and web technologies (ICADIWT 2014) (pp. 232–238). IEEE.

Korycki, L., & Krawczyk, B. (2021). Low-dimensional representation learning from imbalanced data streams. *PAKDD (1). Lecture notes in computer science* (Vol. 12712, pp. 629–641). Springer.

Koziarski, M., Bellinger, C., & Wozniak, M. (2021). RB-CCR: Radial-based combined cleaning and resampling algorithm for imbalanced data classification. *Machine Learning, 110*(11), 3059–3093.

Koziarski, M., Krawczyk, B., & Wozniak, M. (2019). Radial-based oversampling for noisy imbalanced data classification. *Neurocomputing, 343*, 19–33.

Koziarski, M., & Wozniak, M. (2017). CCR: A combined cleaning and resampling algorithm for imbalanced data classification. *International Journal of Applied Mathematics and Computer Science, 27*(4), 727–736.

Kriegel, H., Kröger, P., Schubert, E., Zimek, A. (2009). Loop: Local outlier probabilities. In CIKM (pp. 1649–1652). ACM.

Ksieniewicz, P. (2019). Combining random subspace approach with smote oversampling for imbalanced data classification. *HAIS. Lecture notes in computer science* (Vol. 11734, pp. 660–673). Cham: Springer.

Kubat, M., & Matwin, S. (1997). Addressing the curse of imbalanced training sets: One-sided selection. *ICML* (pp. 179–186). Citeseer.

Lazarevic, A., Kumar, V. (2005). Feature bagging for outlier detection. In KDD (pp. 157–166). ACM.

Lemaitre, G., Nogueira, F., & Aridas, C. K. (2017). Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning. *The Journal of Machine Learning Research, 18*, 17–1175.

Li, Z., Zhao, Y., Botta, N., Ionescu, C., Hu, X. (2020). COPOD: Copula-based outlier detection. In ICDM (pp. 1118–1123). IEEE.

Liu, F. T., Ting, K. M., Zhou, Z.(2008). Isolation forest. In ICDM (pp. 413–422). IEEE Computer Society.

Loureiro, A., Torgo, L., Soares, C. (2004). Outlier detection using clustering methods: A data cleaning application. In Proceedings of KDNet symposium on knowledge-based systems for the public sector. Springer Bonn.

Makki, S., Assaghir, Z., Taher, Y., Haque, R., Hacid, M., & Zeineddine, H. (2019). An experimental study with imbalanced classification approaches for credit card fraud detection. *IEEE Access, 7*, 93010–93022.

Moniz, N., & Cerqueira, V. (2021). Automated imbalanced classification via meta-learning. *Expert Systems with Applications, 178*, 115011.

Naseriparsa, M., Kashani, M. M. R. (2014). Combination of PCA with SMOTE resampling to boost the prediction rate in lung cancer dataset. CoRR **abs/1403.1949**

Nguyen, H. M., Cooper, E. W., & Kamei, K. (2011). Borderline over-sampling for imbalanced data classification. *International Journal of Knowledge Engineering and Soft Data, 3*(1), 4–21.

Niculescu-Mizil, A., Caruana, R. (2005). Predicting good probabilities with supervised learning. In ICML. ACM international conference proceeding series (vol. 119, pp. 625–632). ACM.

Padmaja, T.M., Dhulipalla, N., Bapi, R.S., Krishna, P.R.(2007). Unbalanced data classification using extreme outlier elimination and sampling techniques for fraud detection. In: 15th International Conference on Advanced Computing and Communications (ADCOM 2007) (pp. 511–516). IEEE.

Pearson, K. (1901). LIII. On lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science, 2*(11), 559–572.

Pevný, T. (2016). Loda: Lightweight on-line detector of anomalies. *Machine Learning, 102*(2), 275–304.

Pokrajac, D., Reljin, N., Pejcic, N., Lazarevic, A. (2008). Incremental connectivity-based outlier factor algorithm. In: BCS International Academy Conference (pp. 211–224). British Computer Society.

Prokhorenkova, L. O., Gusev, G., Vorobev, A., Dorogush, A. V., Gulin, A. (2018). Catboost: Unbiased boosting with categorical features. In NeurIPS (pp. 6639–6649).

Rajkomar, A., Oren, E., Chen, K., Dai, A. M., Hajaj, N., Hardt, M., Liu, P. J., Liu, X., Marcus, J., & Sun, M. (2018). Scalable and accurate deep learning with electronic health records. *NPJ Digital Medicine, 1*(1), 1–10.

Rousseeuw, P. J., & van Driessen, K. (1999). A fast algorithm for the minimum covariance determinant estimator. *Technometrics, 41*(3), 212–223.

Roweis, S. T., & Saul, L. K. (2000). Nonlinear dimensionality reduction by locally linear embedding. *Science, 290*(5500), 2323–2326.

Saito, T., & Rehmsmeier, M. (2015). The precision-recall plot is more informative than the roc plot when evaluating binary classifiers on imbalanced datasets. *PloS ONE, 10*(3), 0118432.

Sanguanmak, Y., Hanskunatai, A. (2016). Dbsm: The combination of dbscan and smote for imbalanced data classification. In: 2016 13th International joint conference on computer science and software engineering (JCSSE) (pp. 1–5) . IEEE.

Schölkopf, B., Williamson, R. C., Smola, A. J., Shawe-Taylor, J., Platt, J. C. (1999). Support vector method for novelty detection. In NIPS (pp. 582–588). The MIT Press.

Sharma, S., Bellinger, C., Krawczyk, B., Zaïane, O.R., Japkowicz, N. (2018). Synthetic oversampling with the majority class: A new perspective on handling extreme imbalance. In ICDM (pp. 447–456), IEEE Computer Society.

Shi, C., Li, X., Lv, J., Yin, J., & Mumtaz, I. (2020). Robust geodesic based outlier detection for class imbalance problem. *Pattern Recognition Letters, 131*, 428–434.

Shimauchi, H. (2021). Improving supervised outlier detection by unsupervised representation learning and generative adversarial networks: An extension of extreme gradient boosting outlier detection by gans. In ICISS (pp. 22–27). ACM.

Shwartz-Ziv, R., & Armon, A. (2022). Tabular data: Deep learning is not all you need. *Information Fusion, 81*, 84–90.

Sobhani, P., Viktor, H. L., & Matwin, S. (2014). Learning from imbalanced data using ensemble methods and cluster-based under sampling. *NFMCP. Lecture notes in computer science* (Vol. 8983, pp. 69–83). Springer.

Sotoca, J. M., Sánchez, J., Mollineda, R. A. (2005). A review of data complexity measures and their applicability to pattern classification problems. Actas del III Taller Nacional de Mineria de Datos y Aprendizaje (pp. 77–83). TAMIDA.

Sundarkumar, G. G., & Ravi, V. (2015). A novel hybrid undersampling method for mining unbalanced datasets in banking and insurance. *Engineering Applications of Artificial Intelligence, 37*, 368–377.

Su, X., & Tsai, C. (2011). Outlier detection. *WIREs Data Mining and Knowledge Discovery, 1*(3), 261–268.

Tan, P. (2005). *Introduction to data mining*. Addison-Wesley.

Tarawneh, A. S., Hassanat, A. B. A., Almohammadi, K., Chetverikov, D., & Bellinger, C. (2020). SMOTE-FUNA: Synthetic minority over-sampling technique based on furthest neighbour algorithm. *IEEE Access, 8*, 59069–59082.

Tenenbaum, J. B., Silva, V. D., & Langford, J. C. (2000). A global geometric framework for nonlinear dimensionality reduction. *Science, 290*(5500), 2319–2323.

Tomsett, R., Braines, D., Harborne, D., Preece, A. D., Chakraborty, S. (2018). Interpretable to whom? A role-based model for analyzing interpretable machine learning systems. CoRR **abs/1806.07552**

Torrent, N. L., Visani, G., Bagli, E. (2020). PSD2 explainable AI model for credit scoring. CoRR **abs/2011.10367**

Tran, T. C., Dang, T. K. (2021). Machine learning for prediction of imbalanced data: Credit fraud detection. In IMCOM (pp. 1–7). IEEE.

Van der Maaten, L., & Hinton, G. (2008). Visualizing data using t-SNE. *Journal of Machine Learning Research, 9*(11), 2579–2605.

Veropoulos, K., Campbell, C., Cristianini, N. (1999). Controlling the sensitivity of support vector machines. In Proceedings of the international joint conference on AI (vol. 55, pp. 60). Citeseer.

Wang, X., Liu, X., Matwin, S., & Japkowicz, N. (2014). Applying instance-weighted support vector machines to class imbalanced datasets. *IEEE BigData* (pp. 112–118). IEEE Computer Society.

Wilson, D. L. (1972). Asymptotic properties of nearest neighbor rules using edited data. *IEEE Transactions on Systems, Man, and Cybernetics, 2*(3), 408–421.

Zadrozny, B., Elkan, C. (2002). Transforming classifier scores into accurate multiclass probability estimates. In KDD (pp. 694–699). ACM.

Zhang, Z., Wang, J. (2006). MLLE: Modified locally linear embedding using multiple weights. In NIPS (pp. 1593–1600). MIT Press.

Zhao, Y., Hryniewicki, M. K. (2018). XGBOD: Improving supervised outlier detection with unsupervised representation learning. In IJCNN (pp. 1–8). IEEE

Zhao, Y., Hu, X., Cheng, C., Wang, C., Xiao, C., Wang, Y., Sun, J., Akoglu, L. (2020). SUOD: A scalable unsupervised outlier detection framework. CoRR **abs/2003.05731**

Zhao, Y., Nasrullah, Z., & Li, Z. (2019). Pyod: A python toolbox for scalable outlier detection. *Journal of Machine Learning Research, 20*, 96–1967.