Knowledge and Information Systems (2024) 66:2747–2767
https://doi.org/10.1007/s10115-023-02046-7

REGULAR PAPER

An automated approach for binary classification on imbalanced data

Pedro Marques Vieira[1] · **Fátima Rodrigues**[1,2]

Received: 2 June 2023 / Revised: 5 September 2023 / Accepted: 7 December 2023 /
Published online: 12 January 2024

Abstract
Imbalanced data are present in various business sectors and must be handled with the proper resampling methods and classification algorithms. To handle imbalanced data, there are numerous resampling and learning method combinations; nonetheless, their effective use necessitates specialised knowledge. In this paper, several approaches, ranging from more accessible to more advanced in the domain of data resampling techniques, will be considered to handle imbalanced data. The application developed delivers recommendations of the most suitable combinations of techniques for a specific dataset by extracting and comparing dataset meta-feature values recorded in a knowledge base. It facilitates effortless classification and automates part of the machine learning pipeline with comparable or better results than state-of-the-art solutions and with a much smaller execution time.

Keywords Imbalanced classification · Resampling · Meta-learning · Automated machine learning

1 Introduction

Several current real-world datasets are imbalanced by nature, in that they have one or some classes underrepresented compared to the other class or classes. The class imbalance problem arises in multiple areas, including telecommunication, bioinformatics, fraud detection, and medical diagnosis. The best approach to handle imbalanced data highly depends on the nature of the data. The methods and combination of methods proposed are abundant in various conceivable outcomes, and most times they require specialised knowledge to be used correctly. As such, this paper focuses on an open-ended current problem associated with

✉ Pedro Marques Vieira
1160634@isep.ipp.pt

✉ Fátima Rodrigues
mfc@isep.ipp.pt

1 ISEP, Polytechnic Institute of Porto, Rua Dr. António Bernardino de Almeida, Porto 4249-015, Portugal

2 Interdisciplinary Studies Research Center (ISRC), Porto, Portugal

machine learning (ML) tasks, being a new proposal to automate imbalanced classification, applied to different case study solutions.

Classification algorithms for imbalance scenarios applied without proper data resampling or a cost-sensitive approach, for instance, tend to perform better for well-represented classes and worse for underrepresented classes. In these cases, the underrepresented class tends to be the one with more interest. Multiple strategies have been proposed to address class imbalance problems. However, there is no general guidance on when to use each technique.

In addition, combining different data resampling techniques, classification algorithms, and multiple hyperparameter optimisations makes the possibilities for evaluating the desired solution endless. Thus, a solution to automate and facilitate these imbalanced classification tasks is needed to get better and faster results.

The goal of this study is to develop a system to automatically prepare an imbalanced dataset to be used by a classifier. To accomplish that, this paper includes a review of the state of the art on related solutions, an implementation of the most promising balance techniques, and testing different combinations of them in several public datasets using different classification algorithms. The best balance technique, classification algorithm, and dataset meta-features are recorded in a knowledge base to be recommended for new datasets.

The remainder of this paper is organised as follows: Sect. 2 reviews and discusses existing solutions for imbalanced classification. The developed solution, which includes a learning module and a recommendation module, is described in Sect. 3. The learning module is presented the criteria for dataset selection to be used in the development of the solution, the meta-features extracted from the selected datasets, the evaluation metrics to assess model performance, the resampling and classification algorithms considered, and lastly, the process of selecting the best combinations of resampling and classification algorithms to use in the learning module. In the recommendation module section, the selection process for the best resampling and classification recommendations for a specific dataset is described. Section 4 presents an internal and external evaluation of the recommendation module. The internal evaluation compares the recommendation module results with the best resampling and classification algorithms obtained with the learning module. The external evaluation compares the recommendation module results with the results obtained with an automated ML pipeline framework. The main conclusions and prospects of future work are disclosed in the final section.

2 State of the Art

The research described in this paper sits at the intersection of two major areas: imbalanced classification and automated ML (AutoML). In this section, an overview of both study fields will be provided, along with some AutoML frameworks.

2.1 Imbalanced data

A dataset becomes inherently imbalanced when one class is heavily underrepresented, in their instances, regarding the rest of the classes, in two-class or multi-class datasets. The underrepresented class is designated as the minority class, which has few instances, contrarily to the majority class(es) which has several instances. In this paper, we only focus on the two-class imbalanced learning problem. As such, the minority class is typically the one with the most interest, being represented as the positive one, which corresponds to the class where the correct prediction is more important. The minority class is usually rare, extreme, or unusual in some capacity and faces abundant examples of the majority class. As a result, the need to identify or predict the minority class emphasises how difficult this problem is. The imbalanced ratio of a dataset can be defined as Eq. 1 [1], where N_- and N_+ are cardinalities of the minority and the majority classes, respectively.

$$IR = \frac{N_-}{N_+} \tag{1}$$

However, this ratio can also be expressed, for example, in (1:50), which means that for every one example in the minority class, there are fifty examples in the majority class.

This imbalance property can be categorised into a slight and a severe imbalance [2]. The former applies when the distribution of examples in the training dataset is uneven by a small margin, for example, a distribution of (2:3), and the latter applies when the distribution of examples in the training dataset is uneven by a large margin, such as (1:100) or more. A slight imbalance of the classes is often not a problem because predictive modelling can be achieved without degradation of results [3]. This can happen because, sometimes, the less represented classes are not the most relevant ones, depending on the aim of the work, or when the classes are well separated [1].

2.2 Strategies for handling imbalanced data

Imbalanced learning has been receiving plenty of scientific attention, partly due to its utility in real-world applications. As a result, numerous authors have thoroughly investigated the topic. General surveys of the area can be found in the works [2–5]. The existing approaches to learning under imbalanced domains are divided into four main categories: data pre-processing, special-purpose learning methods, prediction post-processing, and hybrid methods [4]. In this paper, we shall focus on the combination of data balancing methods with classification algorithms.

Data balancing techniques can be divided into weighing the data space when using cost-sensitive procedures or distribution adjustments when resampling the data. This research focuses on distribution adjustment strategies that alter data distribution to more accurately reflect the cases that are more important but underrepresented. Consequently, distribution change and more specifically data sampling algorithms change the composition of the training dataset to improve the performance of a standard ML algorithm on an imbalanced classification problem [3].

Data oversampling involves duplicating examples several times of the minority class or synthesising new examples from the minority class from existing examples. Examples include the synthetic minority oversampling technique (SMOTE), adaptive synthetic sampling approach (ADASYN), borderline SMOTE, SVM SMOTE, and k-means SMOTE. Fernandez et al [3] present an overview of concepts based on the SMOTE algorithm [6].

Data undersampling involves deleting examples from the majority class, such as randomly or using an algorithm to carefully choose which examples to delete [3]. Algorithm examples are random undersampling, condensed nearest neighbour, Tomek links, edited nearest neighbours, neighbourhood cleaning rule, and one-sided selection [7].

Additionally, multiple oversampling and undersampling approaches can be combined. Examples can be SMOTE and random undersampling, SMOTE and Tomek links, and SMOTE and edited nearest neighbours [7]. When applying undersampling there is a risk of losing important cases, and when applying oversampling there is a risk of overfitting because of the replication of certain cases.

For a classification algorithm, the optimal resampling methods are different for different imbalance datasets. Given an imbalanced dataset, the best resampling method is also different when different classification algorithms are applied [8]. Therefore, the selection of the resampling method is related to the classification algorithm as well as the data characteristics.

2.3 Automated Machine Learning

Knowledge Discovery from Data (KDD) is a multi-step process that uses algorithms for each step, including data cleaning, data pre-processing like data labelling, handling imbalanced classes, and feature selection. Next, one or more ML algorithms are trained on the data, followed by knowledge evaluation and refinement. All of these steps are repeated numerous times [9]. Given the variety of KDD tasks and the abundance of ML algorithms, one major challenge is how to choose the best algorithms among the many candidate algorithms that are available for each one of the KDD steps.

The process of automated algorithm selection for each step of the KDD process has received a lot of attention, originating a new research area—Automated Machine Learning (AutoML). AutoML aims to improve the current way of building ML applications by automation [10]. Its key objectives are to reduce the amount of time and resources needed to develop accurate prediction models, support the early implementation of the best solutions, and save time and resources without sacrificing model accuracy. Numerous authors have thoroughly covered the subject of AutoML including high-level overviews [11], to specific issues such as pipeline creation [12], meta-learning [13], and empirical benchmarks of various techniques [10].

In AutoML, methods that are based on meta-learning have shown substantial success concerning algorithm selection. Meta-learning is the process of learning from past experience gathered through the application of learning algorithms to a wide range of datasets, with the end goal of minimising the amount of time required to learn new tasks [13]. The meta-learning strategy is based on learning from dataset characteristics known as dataset meta-features and prior model evaluations to automate algorithm selection. The dataset meta-features permit us to discern what properties the various learning tasks share that make some algorithms more effective at learning them.

Although the goal of AutoML is to automate the complete ML pipeline, the main developments focus only on algorithm selection and hyperparameter optimisation [10] known as a CASH problem [14]. The selection of pre-processing methods is a relatively new but rapidly expanding research area in AutoML. Since pre-processing involves 50% to 80% of the overall KDD process [9], it plays a significant role as it is one of the most expensive steps.

Specific works for pre-processing based on meta-learning include noise filter selection [15] and feature selection [16–18]. Concerning imbalanced learning, to the extent of our knowledge, only two works addressed the automation of imbalanced learning. The first study was conducted by the authors in [8]. They adopt a learning-to-rank approach by selecting the top-K most promising imbalance handling methods using data characteristic measures. The rank of the imbalance handling methods on the dataset is obtained by integrating the ranks of the k neighbours. According to the recommended rank and personal bias, the most appropriate imbalance handling method is picked out. Concerning our proposal, this work has aspects significantly different, such as the optimisation criteria and the meta-learning approach based on a learning-to-rank approach. The other work is the Automated Imbalanced

Table 1 AutoML frameworks

Framework	Cash solver	Search space	Pipeline structure	Pre-Processing	Feature Selection	Data Balancing
Auto-Sklearn	Bayesian	Iterative Algorithms	Semi Fixed	Missing value Imputation	Yes	No
TPOT	Genetic Progr	H2O Pipelines	Variable	limited	Yes	No
H2O AutoML	Grid Search	Scikit-learn Pipelines	Fixed	Normalisation One-hot Encoding	Yes	No

Classification (ATOMIC) method [19] which applies AutoML specifically for imbalanced classification. Like our work, they extract meta-features from the datasets, but differently, they use meta-learning, building a model on the meta-data, which in turn recommends appropriate algorithms according to the learned meta-model. Therefore, the solution is computationally complex and only builds models using the Random Forest learning algorithm.

Also, AutoML frameworks permit building ML pipelines automatically, which mainly involves defining the pipeline structure followed by the selection of algorithms and their hyperparameters. However, the first concern to note is that most of them focus only on some parts of the ML pipeline [20].

For instance, *Auto-sklearn* framework [21] consists of a configuration space, a Bayesian optimiser, a meta-learner, and a model integrator. *Auto-sklearn* uses a Bayesian optimiser to solve the generalised CASH problem and obtain the optimal predictive model. In addition, *Auto-sklearn* integrates two techniques to further improve algorithm performance: first, a meta-learner to obtain the initial configuration space according to prior information to improve the efficiency of the algorithm and, second, a model integrator to combine multiple ML pipelines to improve the algorithm's accuracy. It can do parallelisation on a single computer or in a cluster on a limited time budget.

TPOT [22] is an AutoML framework that optimises pipelines using genetic programming. Using grammar, ML pipelines are expressed as trees where different branches represent distinct pre-processing pipelines. These pipelines are then optimised through evolutionary optimisation. To reduce overfitting that may arise from the large search space, multi-objective optimisation is used to minimise the pipeline complexity while optimising performance [22]. To reduce the search space is also possible by specifying a pipeline template, which dictates the high-level steps in the pipeline.

Finally, there is also *H2O AutoML* [23], an ML framework with APIs in R, Python, Java, and Scala and a web GUI. Its main feature relies on the efficient training of ML algorithms (e.g. GBMs, Random Forests, Deep Neural Networks, and GLMs), yielding a diverse amount of candidate models that are exploited by stacked ensembles to produce a powerful final model. Key aspects of H2O AutoML include its ability to handle missing or categorical data natively, its comprehensive modelling strategy, including powerful stacked ensembles, and the ease with which H2O models can be deployed and used in production environments.

Table 1 presents a summary of the main characteristics of the AutoML frameworks here described.

A rigorous evaluation of these AutoML frameworks can be found in the 2022 OpenML AutoML Benchmark [24].

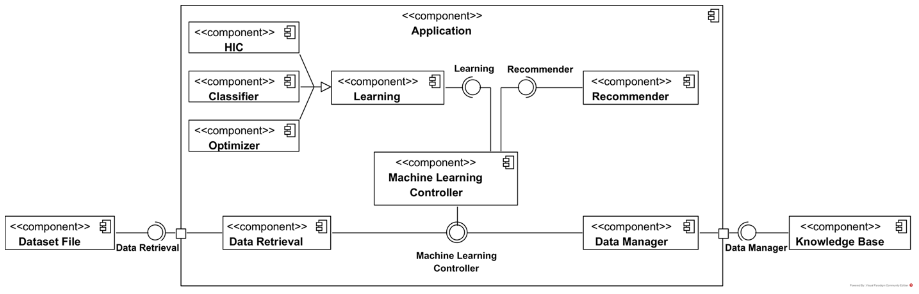


Fig. 1 Component diagram

When analysing all these frameworks there are not any advanced data balancing methods in the context of AutoML, most frameworks offer basic data pre-processing operations and some specific feature selection pipelines, and there are few flexible approaches. In addition, as most frameworks automate pipeline creation, new functionalities are difficult to include, as all of these tools restrict the maximum number of steps. To make AutoML truly available to users, the definition and integration of new facilities are necessary. Moreover, automated imbalanced classification is still in its early days; therefore, the contribution of this paper is to implement a new, easy-to-use application that automates the classification of imbalanced datasets even for less experienced users, mainly because few tools specialise in imbalanced datasets.

3 Developed solution

The application that will be described was built in *Python* and is available, in a *GitHub* repository [25], as free and open-source software, licensed as *GPL 3.0* [26].

The developed application implements two separate but related modules: the learning module, which creates a knowledge base that is used by the recommendation module. The learning module mainly involves the evaluation of balancing and classification algorithms on several datasets, the extraction of meta-features from the datasets, and building a knowledge base with all the information necessary for the recommendation module to suggest the best balancing and classification algorithm for a new dataset without having to run the entire ML pipeline.

To better understand this application, it was envisioned the architecture of the solution expressed as a component diagram in Fig. 1.

A dataset file should be loaded in the application using the data retrieval component that is responsible for reading the dataset file and that is called by the ML controller component. Then, the ML controller component communicates with the learning component, at the early stages of the application, and with the recommendation component, at the late stages of the application. The learning controller is composed of the handling imbalanced classification (HIC), the classifier, and the optimiser components.

This first component applies different techniques to handle imbalanced classification, primarily in the pre-processing stage of the ML pipeline. The classifier component should select the most appropriate classification algorithm for the balanced dataset, and then the optimiser component improves the selected classifier by optimising its parameters. When the best combination of resampling and classification algorithms is found, the ML controller component uses the data manager component that is responsible for writing to the knowledge base all the information concerning this ML pipeline.

3.1 Learning module

The ultimate goal of the learning module is to build a knowledge base with the best features necessary for the recommendation module to suggest the most performing resampling and classification algorithms to process an unbalanced binary dataset. For that, it will be described next: the datasets selected, the dataset meta-features extracted, the evaluation metrics, and the resampling and classification algorithms used.

3.1.1 Datasets

Several imbalanced datasets were chosen from different business domains. The aim is to always choose publicly available datasets without needing to do specific data cleaning tasks before using them. In addition, it was also ensured to have a different ratio of proportions of imbalanced data across the diverse datasets.

Initially, it was analysed several candidate datasets from websites like *UCI Machine Learning Repository* [27], *KEEL—Knowledge Extraction based on Evolutionary Learning* [28], *OpenML*, *Kaggle* [29], and *Google Dataset Search* [30]. Then, it was selected to work with *KEEL* website because it listed the diverse datasets by the imbalanced ratio in an organised manner with key information. Afterwards, it was also selected to work with *OpenML* since it provides plenty of datasets to choose from and has an easy-to-use and well-documented API (application programming interface) [31].

At the time of this paper's development, the *OpenML API* provided 125 datasets when filtering the datasets that have an active status, for binary classification problems, with the number of instances (rows) between 200 and 10000, the number of features (columns) less than 500 and with an imbalance ratio above 2. Of these 125 datasets, some datasets were repeated since they have different versions of the same dataset; in this case, it was selected the most recent one, discarding the older ones. Other datasets were not possible to use because it was not conceivable to provide a decent enough evaluation metrics score. They needed major individual data pre-processing tasks that were not the point of this application to make. From the 125 initial datasets provided by the *OpenML API*, 53 datasets were used. For the same criteria selection, it was also selected 12 datasets from the *KEEL* website, getting a total of 65 datasets to be used. For these 65 datasets, it was found that the imbalanced ratio ranges from 1.820 (minimum) to 85.880 (maximum), averaging 14.501 with a standard deviation of 19.301.

3.1.2 Dataset meta-features

Meta-features for imbalanced classification refer to characteristics or properties of datasets that can provide insights into their level of class imbalance and potential challenges when applying ML algorithms to them. These features are often used to pre-screen or analyse datasets before selecting an appropriate resampling and classification algorithm for an imbalanced classification task. The work [32] provides an excellent survey and evaluation of dataset meta-features for classification tasks that are organised in the following taxonomy:

- *complexity*: estimate the difficulty in separating the data points into their expected classes.

- *concept*: estimate the variability of class labels among examples and the density of the examples.
- *general*: general information related to the dataset, also known as simple measures, such as the number of instances, attributes, and classes.
- *itemset*: compute the correlation between binary attributes.
- *landmarking*: performance of simple and efficient learning algorithms.
- *model-based*: measures designed to extract characteristics from simple ML models.
- *statistics*: standard statistical measures to describe the numerical properties of data distribution.

The authors also made available an open-source meta-feature extraction library (pymfe library [33]) that we use to extract the meta-features only from the original (not resampled) datasets. All meta-features available in the library were extracted. To increase the expressiveness of the meta-features, for those represented by multiple values, we compute the *average*, the *standard deviation*, the *kurtosis*, and the *skewness*. Other meta-features, like the "*c2*" meta-feature of the group *complexity*, which is the value of the imbalance ratio, are solely represented by a scalar. Some of them, like the "*cov*" meta-feature of the group *statistics*, which is the absolute value of the covariance of distinct dataset attribute pairs, are already expressed using a summary function.

3.1.3 Evaluation metrics

The choice of performance metrics is crucial to properly evaluating the effectiveness of a prediction model. Several performance metrics extensively used in balanced domains cannot be applied to the imbalanced case since the use of the majority class in the metric could lead to a misleading evaluation of performance [4]. A well-known example is the accuracy paradox when a high value of accuracy does not correspond to a high-quality model because the model is skewed to the majority class and can mask the obtained results [2].

Choosing an appropriate metric is particularly difficult for imbalanced classification problems because most of the standard metrics that are widely used to evaluate classification models assume a balanced class distribution and do not consider that prediction errors may have different importance. Imbalanced classification problems typically consider the minority class more important than the majority class; as such, performance metrics must focus on the minority class, which is a challenge because the minority class lacks the observations required to effectively test the model. So, when working with imbalanced domains, different evaluation metrics are important to use to achieve a more rigorous evaluation [2].

To evaluate our proposal, we have adopted standard metrics that are more appropriate for imbalanced domains [2, 34]. According to the literature, we have selected 5 evaluation metrics: *Balanced Accuracy*, *F1 score*, *ROC AUC*, *Geometric Mean*, and *Cohen's Kappa*. These evaluation metrics are defined based on the confusion matrix as shown in Table 2. TP and TN denote the number of positive and negative examples that are classified correctly, while FN and FP denote the number of misclassified positive and negative examples, respectively. By convention, the class label of the minority class is positive, and the class label of the majority class is negative.

The most intuitive metric obtained from the confusion matrix is accuracy, which represents the ratio of correctly predicted instances among all instances in the dataset. As already referred this metric is sensitive to imbalanced data as it gives an over-optimistic estimation over the majority class.

Table 2 Confusion matrix

	Predicted Positive	Predicted Negative
Actual Positive	True Positive (TP)	False Negative (FN)
Actual Negative	False Positive (FP)	True Negative (TN)

The true positive rate (TPR), also known as recall or sensitivity, can be understood as the probability that an observed positive instance is classified as positive by the ML classifier. The true negative rate (TNR), or specificity, is the proportion of negative instances that are correctly predicted. Another useful metric is *Precision*, which can be considered the probability of success when an instance is classified as positive. They are given by the following equations:

$$TPR = \frac{TP}{TP + FN} \tag{2}$$

$$TNR = \frac{TN}{TN + FN} \tag{3}$$

$$Precision = \frac{TP}{TP + FP} \tag{4}$$

These metrics are individually insufficient because none of them takes into consideration the entire confusion matrix or all the information that the classifier provides, so they only capture a partial perspective of the classifier's performance [35].

The *Balanced Accuracy* (BA) [36] is the arithmetic mean of the TPR and the TNR, that is, the average of positive and negative instances correctly classified. The BA, unlike accuracy, is robust for evaluating classifiers over imbalanced datasets and is given by the following equation:

$$BA = \frac{TPR + TNR}{2} \tag{5}$$

The F1 score [37] is defined as the harmonic mean of precision and recall. This measure does not consider the ratio of negative instances correctly predicted by the ML classifier, so two models with different TNRs have the same F1 score.

$$F1 = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \tag{6}$$

Another metric adequate to handle imbalanced data is the receiver operating characteristic (ROC) curve [38]. It summarises the performance of the classifiers over a range of TPRs (Eq. 2) and false positive rates (FPRs). The FPR is defined by the equation:

$$FPR = \frac{FP}{FP + TN} \tag{7}$$

When evaluating models with various error rates, the ROC curves can determine which proportion of instances will be correctly classified for a given FPR. While ROC curves provide a visual method to determine the effectiveness of a classifier, the area under the ROC curve (AUC) is a performance metric obtained from ROC that can be used to compare classifiers [39]. It is defined as the proportion of the unit square under the ROC curve. Thus, it takes values in the range [0, 1].

$$ROC\_AUC = \int_0^1 TPR(FPR^{-1}(x))\,\mathrm{d}x \tag{8}$$

Another used metric is *Geometric Mean* (GM) [40] which represents class-wise weighted accuracy rates and is defined as the geometric mean of sensitivity and specificity:

$$GM = \sqrt{TPR \times TNR} \tag{9}$$

Finally, Cohen's Kappa (K) [41] is the measure of the agreement between the model predictions and the actual class values as if they happened by chance and is given by the following equation:

$$K = \frac{2 \times (TP \times TN - FN \times FP)}{(TP + FP) \times (FP + TN) + (TP + FN) \times (FN + TN)} \tag{10}$$

Cohen's Kappa coefficient is more informative than accuracy when working with imbalanced data. However, it is likely to give low values for imbalanced data. The Cohen's Kappa coefficient takes values from -1 to $+1$.

3.1.4 Sampling and classification algorithms

The selection of resampling algorithms aimed to encompass those discussed in state-of-the-art papers on imbalanced binary classification, representing various types of resampling techniques, including undersampling, oversampling, and hybrid sampling.

In total, we tested 19 resampling algorithms, 11 undersampling techniques: *ClusterCentroids*, *CondensedNearestNeighbour*, *EditedNearestNeighbours*, *RepeatedEditedNearestNeighbours*, *AllKNN*, *InstanceHardnessThreshold*, *NearMiss*, *NeighbourhoodCleaningRule*, *OneSidedSelection*, *RandomUnderSampler*, and *TomekLinks*; 6 oversampling techniques: *RandomOverSampler*, *SMOTE*, *ADASYN*, *BorderlineSMOTE*, *KMeansSMOTE*, *SVMSMOTE*; and 2 combinations of over- and undersampling techniques: *SMOTEENN* and *SMOTETomek*.

In regard to classification algorithms, our aim was to select a diverse range of approaches, including two tree-based algorithms (RandomForestClassifier and ExtraTreesClassifier), a probabilistic algorithm (GaussianNB), a generalised linear algorithm (LogisticRegression), a nonparametric algorithm (KNeighborsClassifier), a kernel method (Support Vector Classifier), and five tree-based ensemble learning algorithms (LGBMClassifier, XGBClassifier, AdaBoostClassifier, BaggingClassifier, and GradientBoostingClassifier).

3.1.5 Process of discarding the worst performant combinations

This process started by executing 19 resampling techniques and 1 without any pre-processing technique, combined with 11 classification algorithms, resulting in 220 different combinations of resampling and classification algorithms. The 19 resampling techniques used, as of the time of writing, are all available in the Imbalanced Learn library [42].

Testing 220 combinations of resampling techniques and classification algorithms on 65 datasets would be computationally very expensive, so iteratively, we discarded some of the worst-performing combinations of resampling techniques and classification algorithms.

To do this selection, the 220 combinations of resampling techniques and classification algorithms were first applied to one dataset randomly chosen, which permitted the association of each combination with a *final score*, resulting from the average of the 5 metrics previously presented, and a corresponding ranking position, for example, position 22 from the 220 total combinations. Next, two lists were initialised, one concerning the resampling techniques and the other with the classification algorithms; both lists were ordered from better to worse scores by the ranking position of the resampling technique and classification algorithm, respectively.

Then, when some more datasets were randomly chosen and processed, the various positions of each combination were analysed by ordering them first by the resampling technique and then by the classifier. Next, the combinations with the worst scores, with values above the third quartile (75% to 100%), were discarded for all the processed datasets.

In the first step, after 3 datasets were imported and processed, 5 resampling techniques and 3 classification algorithms were discarded, leaving 120 combinations. The algorithm was iteratively applied to several datasets, randomly chosen in each iteration. After five time steps, a total of 16 resampling techniques and 8 classification algorithms were discarded, for a total of 31 datasets processed. The rest of the datasets imported and processed no longer caused discarding more combinations because it was not found any worse performing resampling technique or classifier based on the previous explanation.

In the end, the remaining combinations were 12 with 4 resampling techniques (3 oversampling techniques: RandomOverSampler, SMOTE, SVMSMOTE, and 1 combination of over- and undersampling techniques: SMOTETomek) and 3 boosted tree algorithms: LGBMClassifier, XGBClassifier, and GradientBoostingClassifier.

3.2 Recommendation module

The objective of this module is to make suggestions for the most effective resampling and classification algorithm combinations to use with a specific imported dataset.

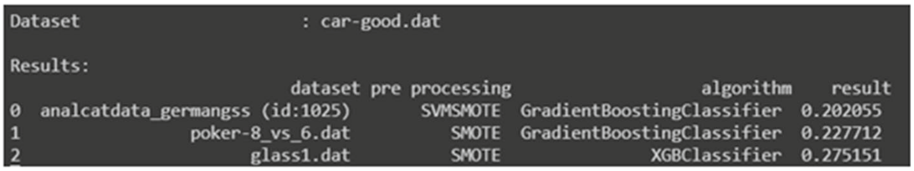
For this, we started to get the best recommendation by developing a multi-classification model using the meta-feature values of each dataset as prediction features and, as a target attribute, the combination of resampling techniques and classification algorithms. However, overfitting occurred due to the complexity of the classifiers and the small size of the training set: 65 instances (the number of datasets available), each with 257 meta-feature values, and 12 different target values to predict. Therefore, we calculated the best recommendations following an instance-based learning approach.

For that, the *Frobenius norm* (the *Euclidean distance* of two vectors) is computed, which, in this case, is the average of all *Euclidean distances* of each meta-feature of the current imported dataset and the meta-features of each of the datasets in the knowledge base. The *Frobenius norm* can be expressed as Eq. 11.

$$\|A\|_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}\left|a_{i,j}\right|^2} \tag{11}$$

This takes into consideration the previously processed 257 meta-features (m), the 65 imported datasets (n), and the values of each meta-feature ($a_{i,j}$).

Next, the three smaller average values are selected, since a smaller value means that those two datasets resemble the most in terms of the features used. By knowing the corresponding datasets, the three combinations of resampling techniques and classification algorithms that are distinct and were recorded as the better performant ones are recommended, in the learning module, for those datasets.



```
Dataset                     : car-good.dat

Results:
                          dataset pre processing                     algorithm    result
0  analcatdata_germangss (id:1025)       SVMSMOTE  GradientBoostingClassifier  0.202055
1               poker-8_vs_6.dat          SMOTE  GradientBoostingClassifier  0.227712
2                     glass1.dat          SMOTE               XGBClassifier  0.275151
```

Fig. 2 GUI recommendations output example

For instance, as illustrated in Fig. 2, submitting the "*car-good.dat*" dataset to the recommendation module finds "*analcatdata_germangss*", "*poker-8_vs_6.dat*", and "*glass1.dat*" as the datasets with the lowest Euclidean distances, 0.202055, 0.227712, and 0.275151, respectively.

For those datasets, the best combinations of resampling techniques and classifiers found by the learning module are (*SVMSMOTE*, *GradientBoostingClassifier*), (*SMOTE*, *GradientBoostingClassifier*), (*SMOTE*, *XGBClassifier*), which are recommended.

4 Solution evaluation

The evaluation of the solution is conducted with two distinct steps, an internal evaluation and an external evaluation. The former is made by analysing and comparing the recommended results, with the results that were acquired by the learning module. The latter is made by analysing and comparing the recommended results with the *TPOT* AutoML framework.

Concerning the datasets chosen to evaluate this application internally and externally, 15 datasets were randomly selected from the initial 65. However, it should be noted that both internal and external evaluations depend on the performance of the recommendation system, and this one works by searching for the datasets closest to the test dataset. So, these 15 test datasets were not considered in the knowledge base of the recommendation system, as this would not make sense since the recommendation module is based on searching for the datasets closest to the test dataset that is intended to find the best techniques to apply.

These 15 test datasets' unbalanced ratios range from 2.307 (the minimum) to 67 (the maximum), with an average of 18.662 and a standard deviation of 21.998. Table 3 displays the datasets, their dimensions, and their imbalance ratio.

The evaluation metrics employed to assess the various solutions are the same as those used in the creation of the knowledge base. Additionally, it was assumed that the minority target class is the most relevant to predict.

The default parameters of all resampling and classification algorithms were used, and for all the executions, it was addressed the guarantee of reproducibility with *random_state*. Also, all the processors of the machine during the cross-validation step were used with *n_jobs*. When it was possible to automatically adjust the class weights inversely proportional to class frequencies, it was used the *class_weight* equal to "*balanced*" mode, or to specify the learning objective function that the dataset is binary.

It was chosen a *Stratified K-Fold* cross-validation with 10 folds, repeated 3 times, with different randomisation in each repetition, which is common practice in imbalance scenarios, to assure a rigorous estimator performance.

Also, all evaluation tasks were executed with the same conditions of the same available local computer resources.

Table 3 Datasets selected to test the application

ID	Dataset	Lines × columns	IR
D1	*dis* (*OpenML ID*:40713)	3772 × 30	64.034
D2	*musk* (*OpenML ID*:1116)	2000 × 100	5.488
D3	*mfeat-fourier* (*OpenML ID*:971)	2000 × 77	9.000
D4	*Satellite* (*OpenML ID*:40900)	5100 × 37	67.000
D5	*arsenic-male-bladder* (*OpenML ID*:947)	5590 × 5	22.292
D6	*analcatdata_apnea2* (*OpenML ID*:765)	475 × 4	6.422
D7	*regime_alimentaire* (*OpenML ID*:42172)	220 × 20	3.744
D8	*page-blocks0.dat*	5473 × 10	8.789
D9	*dgf_test* (*OpenML ID*:42883)	3420 × 5	5.053
D10	*cpu_small* (*OpenML ID*:735)	8190 × 13	2.307
D11	*analcatdata_birthday* (*OpenML ID*:968)	365 × 4	5.837
D12	*optdigits* (*OpenML ID*:980)	5620 × 65	8.825
D13	*kr-vs-k-zero_vs_eight.dat*	1460 × 6	53.074
D14	*analcatdata_lawsuit* (*OpenML ID*:450)	264 × 5	12.895
D15	*JapaneseVowels* (*OpenML ID*:976)	9960 × 15	5.172

4.1 Internal evaluation

Regarding the internal evaluation, the recommendation module will never perform better than the learning module; it may present an equal performance if it returns a combination of resampling and classification algorithms equal to those of the learning module or worse if one of the algorithms it proposes is different. This happens because the learning module tests all potential resampling and classification algorithm combinations before choosing the optimal one. Table 4 presents the combinations of resampling and classifier algorithms obtained with the learning module and with the recommendation module (the first recommended combination of the three combinations available) when executing with those 15 datasets.

As can be seen from Table 4, for only 3 datasets (D2, D5, and D6), the recommendation module gives the same suggestions as the learning module, but 7 recommendations have one of the algorithms, balancing and/or classification, in common with those given by the learning module.

Next, we will quantitatively assess both modules. The performance obtained from the combination of balancing and classification algorithms suggested by the learning module (LM) and the recommendation module (RM) is shown in Table 5, along with the values of the evaluation metrics *Balanced Accuracy* (BA), *F1 Score* (F1), *ROC_AUC* (AUC), *Geometric Mean* (GM), and *Cohen's Kappa* (K).

Concerning imbalance classification evaluation, choosing an appropriate metric is challenging because not all classes or prediction errors are equally important; they depend on the context of the problem. But, in this paper, we do not have a specific problem to analyse but several datasets from different business domains, so we averaged the various metrics to cover different aspects of the model's performance, such as accuracy, precision, recall, and overall agreement, and thus had a more holistic assessment of their performance.

Stratified k-fold cross-validation is a robust technique for assessing the performance of ML models. However, it is important to consider factors such as the magnitude of differences,

Table 4 Resampling and classification algorithms of both modules

	Learning module		Recommendation module	
Dataset	Resampling	Classification	Resampling	Classification
D1	RandOverSampler	GradBoostClassif	RandOverSampler	LGBMClassif
D2	RandOverSampler	XGBClassifier	RandOverSampler	XGBClassifier
D3	SMOTE	GradBoostClassif	SVMSMOTE	GradBoostClassif
D4	SMOTETomek	LGBMClassif	SMOTE	GradBoostClassif
D5	RandOverSampler	LGBMClassif	RandOverSampler	LGBMClassif
D6	RandOverSampler	GradBoostClassif	RandOverSampler	GradBoostClassif
D7	SVMSMOTE	LGBMClassif	SVMSMOTE	XGBClassifier
D8	RandOverSampler	XGBClassifier	SMOTE	LGBMClassif
D9	RandOverSampler	XGBClassifier	SVMSMOTE	LGBMClassif
D10	SMOTETomek	LGBMClassif	SMOTETomek	GradBoostClassif
D11	SVMSMOTE	XGBClassifier	RandOverSampler	LGBMClassif
D12	SVMSMOTE	XGBClassifier	RandOverSampler	GradBoostClassif
D13	RandOverSampler	GradBoostClassif	SVMSMOTE	XGBClassifier
D14	RandOverSampler	LGBMClassif	SVMSMOTE	GradBoostClassif
D15	SVMSMOTE	LGBMClassif	RandOverSampler	GradBoostClassif

Table 5 Learning and recommendation evaluation metrics values

	BA		F1		AUC		GM		K	
	LM	RM	LM	RM	LM	RM	LM	RM	LM	RM
D1	0.868	0.787	0.990	0.995	0.943	0.915	0.852	0.747	0.523	0.614
D2	0.998	0.998	0.996	0.996	1.000	1.000	0.998	0.998	0.996	0.996
D3	0.995	0.990	0.999	0.999	1.000	0.999	0.995	0.990	0.993	0.986
D4	0.875	0.882	0.752	0.672	0.993	0.984	0.860	0.870	0.749	0.666
D5	0.795	0.795	0.636	0.636	0.836	0.836	0.716	0.716	0.625	0.625
D6	0.936	0.936	0.833	0.833	0.972	0.972	0.934	0.934	0.804	0.804
D7	0.949	0.940	0.899	0.876	0.973	0.977	0.947	0.938	0.869	0.840
D8	0.946	0.950	0.883	0.868	0.990	0.992	0.945	0.949	0.870	0.852
D9	0.987	0.987	0.971	0.971	0.999	0.999	0.987	0.987	0.966	0.965
D10	0.916	0.914	0.947	0.943	0.979	0.976	0.916	0.913	0.827	0.816
D11	0.860	0.800	0.932	0.937	0.936	0.944	0.851	0.778	0.619	0.576
D12	0.979	0.982	0.997	0.996	0.999	0.999	0.979	0.982	0.971	0.960
D13	0.999	0.980	0.970	0.947	1.000	0.998	0.999	0.977	0.969	0.945
D14	0.970	0.966	0.916	0.873	0.993	0.991	0.965	0.962	0.909	0.863
D15	0.989	0.978	0.995	0.987	1.000	0.998	0.989	0.978	0.972	0.925

variability across folds, and the practical significance of the results that a statistical test can give us. For comparing more rigorously the performance of both modules, the Wilcoxon signed-rank test is carried out on the paired *final score* from the learning and recommendation modules to validate their results further and determine whether there exists a significant difference among them. The null hypothesis of the test is that the median difference between

Table 6 Wilcoxon signed-rank test results

	Final Score			Hypothesis
Dataset	LM	RM	p_value	(α = 0.05)
D1	0.835	0.812	0.0672	Not rejected
D2	0.998	0.998	–	–
D3	0.996	0.993	0.0192	Rejected
D4	0.846	0.815	0.0120	Rejected
D5	0.722	0.722	–	–
D6	0.896	0.896	–	–
D7	0.927	0.914	0.2395	Not rejected
D8	0.927	0.922	0.0743	Not rejected
D9	0.982	0.982	0.9089	Not rejected
D10	0.917	0.912	0.0014	Rejected
D11	0.840	0.807	0.0173	Rejected
D12	0.985	0.984	0.2173	Not rejected
D13	0.987	0.969	0.0679	Not rejected
D14	0.951	0.931	0.0716	Not rejected
D15	0.989	0.973	1.7149E–06	Rejected

the paired scores is zero, while the alternative hypothesis is that there is a significant difference. The choice of the Wilcoxon signed-rank test is because it does not assume a specific distribution for the data and is suitable for nonparametric analysis, making it useful when dealing with performance metrics that might not follow a normal distribution.

Table 6 displays the *final score* for the LM and RM together with the Wilcoxon signed-rank test findings. As already explained, the LM outperforms the RM in all datasets. However, the Wilcoxon signed-rank test's p-value only rejects the null hypothesis for 5 datasets, meaning that the RM performs similarly to the LM for the remaining 10 datasets. The RM's performance is thus around 67% similar to that of the LM.

Concerning the execution time for all 15 datasets, the execution of the RM was accomplished in 1073s and the LM in 8237 s. Thus, for these 15 datasets, the RM time was approximately 8 times smaller or faster than the LM time. This was expected because it is usually faster to execute one instance-based learning algorithm on some meta-feature values than to execute several combinations of resampling techniques and classification algorithms.

4.2 External evaluation

In the external evaluation, we begin by exploring the three AutoML frameworks previously analysed in the State of the Art Section to select one that implements a similar ML pipeline to execute for these 15 datasets. We selected the *TPOT* because this framework is open source and permits defining parameters that assure test conditions like those defined by our application. To guarantee for this framework identical execution time values as the RM, we tried greater or smaller values with a try-error approach for several *TPOT* parameters. Regarding the maximum time that the *TPOT* framework can optimise the pipeline, we define a closer value to the maximum time that the LM achieved with one of the 15 test datasets. Additionally, it used the same cross-validation technique as the developed application.

Table 7 Resampling and classification algorithms

	Recommendation module		TPOT framework
Dataset	Resampling	Classification	Classification
D1	RandOverSampler	LGBMClassif	RandomForestClassif
D2	RandOverSampler	XGBClassifier	GaussianNB
D3	SVMSMOTE	GradBoostClassif	GaussianNB
D4	SMOTE	GradBoostClassif	RandomForestClassif
D5	RandOverSampler	LGBMClassif	GaussianNB
D6	RandOverSampler	GradBoostClassif	SGDClassif
D7	SVMSMOTE	XGBClassifier	RandomForestClassif
D8	SMOTE	LGBMClassif	RandomForestClassif
D9	SVMSMOTE	LGBMClassif	RandomForestClassif
D10	SMOTETomek	GradBoostClassif	RandomForestClassif
D11	RandOverSampler	LGBMClassif	RandomForestClassif
D12	RandOverSampler	GradBoostClassif	RandomForestClassif
D13	SVMSMOTE	XGBClassifier	RandomForestClassif
D14	SVMSMOTE	GradBoostClassif	GaussianNB
D15	RandOverSampler	GradBoostClassif	RandomForestClassif

Similar to how it was done in the internal evaluation, Table 7 depicts the algorithms used by the RM and the classification algorithm used by the TPOT framework, as this framework does not apply balancing functions. Additionally, Table 8 compares the evaluation metrics values for the 15 datasets acquired using the RM to those obtained using the TPOT framework (TF). The results of the Wilcoxon signed-rank test are shown in Table 9 along with the *final score* for the RM and the TF.

As can be observed from Table 9, the *final score* of RM is higher than the final score of TF for all datasets, demonstrating the superiority of RM. There is a statistically significant difference between these two solutions, as shown by the Wilcoxon signed-rank test's p-value, which rejects the null hypothesis for all datasets but one. The RM outperformed the TF in 14 out of the 15 datasets examined or 93% of the datasets. This emphasises the importance of balancing procedures.

Concerning the execution time for all 15 datasets, the execution of the RM was accomplished in 1073 s and the TF in 1381 s. Thus, for these 15 datasets, the RM time was 29% smaller or faster than the TF time execution.

5 Conclusions

The application here described can deliver recommendations of suited combinations of resampling and classification algorithms to binary imbalanced datasets, therefore automating this step in the ML pipeline and thus reducing the human effort placed in building accurate predictive models.

Such tasks are complicated and time-consuming because they require testing a significant number of possible solutions. The proposed application takes advantage of solutions already tested with previous datasets and provides recommendations for a new dataset by choosing

Table 8 Recommendation module and TPOT framework evaluation metrics values

	BA		F1		AUC		GM		K	
	RM	TPOT	RM	TF	RM	TF	RM	TF	RM	TF
D1	0.787	0.677	0.995	0.993	0.915	0.677	0.747	0.549	0.614	0.416
D2	0.998	0.996	0.996	0.978	1.000	0.996	0.998	0.996	0.996	0.974
D3	0.990	0.974	0.999	0.997	0.999	0.974	0.990	0.974	0.986	0.966
D4	0.882	0.809	0.672	0.740	0.984	0.809	0.870	0.778	0.666	0.737
D5	0.795	0.771	0.636	0.630	0.836	0.771	0.716	0.673	0.625	0.623
D6	0.936	0.667	0.833	0.385	0.972	0.667	0.934	0.463	0.804	0.349
D7	0.940	0.827	0.876	0.722	0.977	0.827	0.938	0.785	0.840	0.669
D8	0.950	0.914	0.868	0.871	0.992	0.914	0.949	0.910	0.852	0.858
D9	0.987	0.975	0.971	0.949	0.999	0.975	0.987	0.974	0.965	0.939
D10	0.914	0.903	0.943	0.944	0.976	0.903	0.913	0.901	0.816	0.813
D11	0.800	0.725	0.937	0.926	0.944	0.725	0.778	0.664	0.576	0.456
D12	0.982	0.971	0.996	0.995	0.999	0.971	0.982	0.971	0.960	0.953
D13	0.980	0.903	0.947	0.862	0.998	0.903	0.977	0.884	0.945	0.861
D14	0.966	0.813	0.873	0.634	0.991	0.813	0.962	0.718	0.863	0.612
D15	0.978	0.969	0.987	0.991	0.998	0.969	0.978	0.969	0.925	0.945

Table 9 Wilcoxon signed-rank test results

	Final Score			Hypothesis
Dataset	RM	TF	p_value	(α = 0.05)
D1	0.812	0.662	7.3251E-06	Rejected
D2	0.998	0.988	0.0021	Rejected
D3	0.993	0.977	4.2736E-05	Rejected
D4	0.815	0.775	0.0101	Rejected
D5	0.722	0.694	0.2878	Not rejected
D6	0.896	0.506	1.7333E-06	Rejected
D7	0.914	0.766	0.0003	Rejected
D8	0.922	0.893	5.5050E-05	Rejected
D9	0.982	0.962	2.5356E-06	Rejected
D10	0.912	0.893	1.8852E-06	Rejected
D11	0.807	0.699	3.4052E-05	Rejected
D12	0.984	0.972	2.6948E-06	Rejected
D13	0.969	0.883	0.0105	Rejected
D14	0.931	0.718	0.0002	Rejected
D15	0.973	0.969	0.0007	Rejected

the most similar datasets in terms of meta-features, thus helping to automate the development of efficient solutions to imbalance binary classification problems.

According to the outcomes of the various balancing and classification algorithms that were tested, oversampling in conjunction with boosted trees is a useful strategy for dealing with imbalanced classification.

Additionally, appropriate evaluation metrics were used to compare the various balance and classification combinations proposed with the best possible solution as well as with an AutoML solution. Due to the absence of balancing mechanisms, the AutoML solution was only partially successful. The analysis revealed that the AutoML solutions have not yet concentrated on dealing with imbalanced classification issues. Consequently, this paper is a contribution to the state of the art, despite some limitations and the need for additional research, as will be addressed in the next section.

5.1 Limitations and future work

While the objectives were accomplished, this application might still use some improvements. First, it would be beneficial to evaluate the performance of the evaluation metrics used, first in terms of the measures' consistency with one another and then in terms of the metrics' level of discriminant, to compare the proposed solutions more effectively.

Furthermore, it can also be applied a meta-feature selection like *principal component analysis* to the extracted meta-features to optimise the instance-based learning search of similar datasets.

Additionally, it is imperative to add new datasets to the knowledge base, which will certainly improve the application's outcomes. Further, to accommodate more balancing and classification algorithms, the requirements for discarding the worst-performing combinations must be relaxed.

Finally, in the future, this application should be extended to operate with multi-class classification problems.

Author Contributions Pedro Marques Vieira helped in application implementation, original manuscript preparation, and writing. Fátima Rodrigues contributed to conceptualisation, supervision, and review/editing.

Funding Open access funding provided by FCT|FCCN (b-on).

Availability of supporting data The data used in this study are openly available from public sources as described in the text. The code developed is freely available at GitHub and licensed as *GPL 3.0*.

Declarations

Conflict of interest The authors have no competing interests to declare.

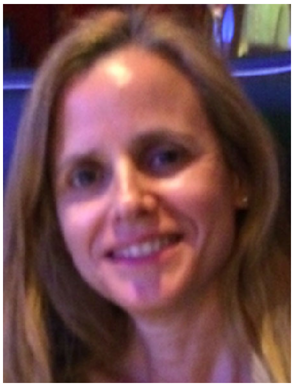
References

1. Lango M (2019) Tackling the problem of class imbalance in multi-class sentiment classification: an experimental study. Found Comput Decis Sci 44(2):151–178. https://doi.org/10.2478/fcds-2019-0009

2. Krawczyk B (2016) Learning from imbalanced data: open challenges and future directions. Prog Artif Intell 5(4):221–232. https://doi.org/10.1007/s13748-016-0094-0
3. Fernández A, García S, Galar M, Prati RC, Krawczyk B, Herrera F (2018) Learning from imbalanced data sets, vol 10. Springer. https://doi.org/10.1007/978-3-319-98074-4
4. Branco P, Torgo L, Ribeiro RP (2016) A survey of predictive modeling on imbalanced domains. ACM Comput Surv (CSUR) 49(2):1–50. https://doi.org/10.1145/2907070
5. Haixiang G, Yijing L, Shang J, Mingyun G, Yuanyue H, Bing G (2017) Learning from class-imbalanced data: review of methods and applications. Expert Syst Appl 73:220–239. https://doi.org/10.1016/j.eswa.2016.12.035
6. Chawla NV, Bowyer KW, Hall LO, Kegelmeyer WP (2002) Smote: synthetic minority over-sampling technique. J Artif Intell Res (JAIR) 16:321–357. https://doi.org/10.1613/jair.953
7. Chaplot A, Choudhary N, Jain K (2019) A review on data level approaches for managing imbalanced classification problem. Int J Sci Res Sci Eng Technol 6(2):91-97. https://doi.org/10.32628/IJSRSET196225 https://doi.org/10.32628/IJSRSET196225 https://doi.org/10.32628/IJSRSET196225
8. Zhang X, Li R, Zhang B, Yang Y, Guo J, Ji X (2019) An instance-based learning recommendation algorithm of imbalance handling methods. Appl Math Comput 351:204–218. https://doi.org/10.1016/j.amc.2018.12.020
9. Fayyad U, Piatetsky-Shapiro G, Smyth P (1996) From data mining to knowledge discovery in databases. AI Mag 17(3):37. https://doi.org/10.1609/aimag.v17i3.1230
10. Zöller MA, Huber MF (2021) Benchmark and survey of automated machine learning frameworks. J Artif Intell Res. https://doi.org/10.1613/jair.1.11854
11. Tuggener L, Amirian M, Rombach K, Lörwald S, Varlet A, Westermann C, Stadelmann T (2019) Automated machine learning in practice: state of the art and recent results. In: 6th Swiss Conference on Data Science (SDS), pp 31-36. IEEE. https://doi.org/10.21256/zhaw-3156
12. Hutter F, Kotthoff L, Vanschoren J (2019) Automated machine learning: methods, systems, challenges. Springer Nature, New York. https://doi.org/10.1007/978-3-030-05318-5
13. Vanschoren J (2018) Meta-learning: a survey. https://doi.org/10.48550/arXiv.1810.03548
14. Thornton C, Hutter F, Hoos H, Leyton-Brown K (2013) Auto-WEKA: combined selection and hyperparameter optimization of classification algorithms. In: ACM International Conference on Knowledge Discovery and Data Mining, pp 847–855. https://doi.org/10.1145/2487575.2487629
15. Garcia L, Carvalho A, Lorena A (2016) Noise detection in the meta-learning level. Neurocomputing 176:14–25. https://doi.org/10.1016/j.neucom.2014.12.100
16. Parmezan AR, Lee HD, Wu FC (2017) Metalearning for choosing feature selection algorithms in data mining: proposal of a new framework. Expert Syst Appl 75:1–24. https://doi.org/10.1016/j.eswa.2017.01.013
17. Shen Z, Chen X, Garibaldi JM (2020) A novel meta learning framework for feature selection using data synthesis and fuzzy similarity. In: IEEE international conference on fuzzy systems (FUZZ-IEEE), pp 1–8. https://doi.org/10.1109/FUZZ48607.2020.9177769
18. Khan I, Zhang X, Ayyasamy RK, Ali R (2023) AutoFe-Sel: a meta-learning based methodology for recommending feature subset selection algorithms. KSII Trans Internet Inform Syst. https://doi.org/10.3837/tiis.2023.07.002
19. Moniz N, Cerqueira V. Automated imbalanced classification via meta-learning. Expert Syst Appl 178:115011 .https://doi.org/10.1016/j.eswa.2021.115011
20. He X, Zhao K, Chu X (2021) AutoML: a survey of the state-of-the-art. Knowl-Based Syst 212:106622. https://doi.org/10.1016/j.knosys.2020.106622
21. M. Feurer, K. Eggensperger, S. Falkner, M. Lindauer, and F. Hutter, 'Auto-Sklearn 2.0: Hands-free AutoML via Meta-Learning', 2020, http://arxiv.org/abs/2007.04074 accessed: Feb. 13, 2022
22. Olson, R.S., Bartley, N., Urbanowicz, R.J. and Moore, J.H., Evaluation of a tree-based pipeline optimisation tool for automating data science. In Proceedings of the genetic and evolutionary computation conference pp. 485-492, 2016. https://doi.org/10.1145/2908812.2908918
23. LeDell E, Poirier S (2020) H2o automl: Scalable automatic machine learning. In Proceedings of the AutoML Workshop at ICML (Vol. 2020). ICML. https://www.automl.org/wp-content/uploads/2020/07/AutoML_2020_paper_61.pdf
24. Gijsbers P, Bueno M L, Coors S, LeDell E, Poirier S, Thomas J, Vanschoren J (2022). Amlb: an automl benchmark. arXiv preprint. https://doi.org/10.48550/arXiv.2207.12560
25. P. Vieira, PedroVieira1160634/automated-imbalanced-classification: Automated Imbalanced Classification. https://github.com/PedroVieira1160634/automated-imbalanced-classification accessed Sep. 10, 2022
26. GNU General Public License v3.0 - Project GNU - Free Software Foundation https://www.gnu.org/licenses/gpl-3.0.html accessed Sep. 10, 2022

27. UCI Machine Learning Repository https://archive.ics.uci.edu/ accessed Aug. 01, 2023
28. KEEL: A software tool to assess evolutionary algorithms for Data Mining problems (regression, classification, clustering, pattern mining and so on) https://sci2s.ugr.es/keel/datasets.php accessed Feb. 14, 2022
29. Find Open Datasets and Machine Learning Projects - Kaggle https://www.kaggle.com/datasets accessed Feb. 14, 2022
30. Dataset Search https://datasetsearch.research.google.com/ accessed Feb. 14, 2022
31. OpenML APIs - OpenML Documentation https://docs.openml.org/APIs/ accessed Jul. 30, 2022
32. Rivolli A, Garcia L P, Soares C, Vanschoren J, Carvalho A C (2018) Characterizing classification datasets: a study of meta-features for meta-learning. arXiv preprint. https://doi.org/10.48550/arXiv.1808.10406
33. The PyMFE example gallery – pymfe 0.4.1 documentation https://pymfe.readthedocs.io/en/latest/auto_examples/index.html accessed Aug. 20, 2022
34. Gaudreault J G, Branco P, Gama J (2021) An analysis of performance metrics for imbalanced classification. In International Conference on Discovery Science (pp. 67-77). Cham: Springer International Publishing. https://doi.org/10.1007/978-3-030-88942-5_6
35. De Diego IM, Redondo AR, Fernández RR, Navarro J, Moguerza JM (2022) General Performance Score for classification problems. Appl Intell 52(10):12049–12063. https://doi.org/10.1007/s10489-021-03041-7
36. Brodersen K H, Ong C S, Stephan K E, Buhmann J M (2010) The balanced accuracy and its posterior distribution. In 20th international conference on pattern recognition (pp. 3121-3124). IEEE. https://doi.org/10.1109/ICPR.2010.764
37. Ferri C, Hernández-Orallo J, Modroiu R (2009) An experimental comparison of performance measures for classification. Pattern Recogn Lett 30(1):27–38. https://doi.org/10.1016/j.patrec.2008.08.010
38. Fawcett T (2006) An introduction to ROC analysis. Pattern Recogn Lett 27(8):861–874. https://doi.org/10.1016/j.patrec.2005.10.010
39. Bradley AP (1997) The use of the area under the ROC curve in the evaluation of machine learning algorithms. Pattern Recogn 30(7):1145–1159. https://doi.org/10.1016/S0031-3203(96)00142-2
40. Tharwat A (2020) Classification assessment methods. Applied computing and informatics 17(1):168–192. https://doi.org/10.1016/j.aci.2018.08.003
41. McHugh ML. Interrater reliability: the kappa statistic. Biochem Med (Zagreb). 2012;22(3):276-82. PMID: 23092060; PMCID: PMC3900052. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3900052/
42. Imbalanced-learn documentation – Version 0.9.1 https://imbalanced-learn.org/stable/ accessed Sep. 10, 2022

Pedro Marques Vieira is a software developer employed at Sistrade Software Consulting in Porto. Sistrade is an information systems company with knowledge in software development and consulting services for different activity areas, including industry companies. He has a degree and master's in computer engineering from ISEP, the Polytechnic Institute of Porto. The thesis of the degree is "Specification and Development of a Contextual Intelligent Network" associated with Sistrade. The master is in the Information and Knowledge Systems branch, where he developed the master's thesis "Automatic Handling of Imbalanced Datasets for Classification". Professor Fátima Rodrigues supervised both theses.



Fátima Rodrigues is currently an associate professor at ISEP, the Polytechnic Institute of Porto, and a researcher in the Interdisciplinary Studies Research Centre (ISRC) at ISEP. Her main skills and expertise are related to business analytics, data science, decision support systems, neural networks, and machine learning. She is the co-author of more than 25 indexed (e.g., ISI, Scopus) publications in international peer-reviewed journals. She has participated in more than seven R&D projects and has supervised four PhD thesis, 35 MSc thesis, and 65 BSc final graduation projects in the area of Intelligent Data Analysis. She has been a regular reviewer of ISI JCR journals such as IEEE Trans. Neural Networks and Learning Systems, Information Sciences, Decision Support Systems, and Data and Knowledge Engineering. Moreover, she has been program committee/reviewer of several international conferences/workshops.