An Efficient Learning Approach to Imbalanced Multinomial Classification

Ani Petkova[1], Borislava Toleva[1], Ivan Ivanov[1, *]
[1]Faculty of Economics and Business Administration, Sofia University St. Kl. Ohridski, Sofia 1113, Bulgaria
Emails: ani.petkova.99@gmail.com; bvrigazova@gmail.com; i_ivanov@feb.uni-sofia.bg

Abstract

The presented methodology provides an innovative way to answer a question that is rarely observed in academic literature: How can complex data issues like multiple class imbalance be solved using the available models in a simple and efficient way? In this approach, observations are modeled without additional preprocessing. Several classification models including Random Forest (RF), Support Vector Machines (SVM), and Decision Tree (DT) are utilized for conducting the classification analysis. The parameters of these models and the cross-validation function are adjusted to each individual set of observations. This approach has not been researched in depth. We test it about class imbalance in the target variable. Our results demonstrate the benefits of the proposed method. First, parameter tuning of ML models can be an effective strategy to handle class imbalance. Second, random shuffling prior to cross validation can be a key to resolving the bias coming from multiclass imbalance. Another important finding is that the best results can be achieved when random shuffling, cross validation and parameter tuning are combined. These findings are key to handling class imbalance in classification. Therefore, this research extends the opportunities to handle class imbalance in a simple, quick, and effective way in cases without adding additional complexity to the model.

Keywords: Multiclass data; Classification; Parameters adjustment; Model evaluation.

1. Introduction

Big data has various applications and various sources. The unstructured data is recorded in a file or in a series of files. Manayika et al. [1] define big data as "an amount of data that surpasses the capabilities of modern technology to process, store and compute them efficiently". This concise definition fully captures the essence of big data and demonstrates their dynamic nature, upon which the dynamics in computer technology for the storage, processing, and public presentation of the data. Usually, a dataset is divided into classes; it can be analysed in the sense of creating an efficient predictive model of the observations within the set. In this article, the number of classes is denoted as m. If m=2, then the set is binary or two-class. If m>2, then the set is multiclass. Often, in a dataset, one or more classes have significantly fewer observations compared to the other observations in the set. Such sets are called imbalanced.

The classification of imbalanced data is one of the most critical challenges faced by modern data analysis. Particularly when combined with other factors of difficulty such as noise presence and overlapping in class distributions, data imbalance can significantly affect the results of classification analysis. Therefore, Branco et al. [2] have proposed adjusted classification metrics targeted at imbalanced data. An alternative approach is to balance classes artificially which is suggested by Koziarski et al. [3] and the standard classification metrics are applied and described by James et al. [4]. The most common balancing technique is the Synthetic Minority Oversampling Technique (SMOTE). SMOTE synthesizes new examples of the minority class in the training set by drawing a line between two adjacent examples. Then, new examples of the minority class are produced as a combination of the two examples. However, some of the data complexity factors impact the performance of existing oversampling

DOI: https://doi.org/10.54216/FPA.180215
Received: July 14, 2024 Revised: October 21, 2024 Accepted: January 06, 2025

strategies, particularly SMOTE and its derivatives considered by Chawla et al. [5], Han et al. [6], Maldonado et al. [7]. This effect is particularly pronounced in a multi-class scenario, where the mutual imbalance of class ratios further complicates matters and it is investigated by some authors [3,8,9,10]. Despite that, most modern research in the field of data imbalance is focused on tasks with binary classification, while their more challenging multiclass counterparts remain relatively underexplored.

A full review of approaches to handle class imbalance can be found in Rezvani et al. [11]. Most approaches to handling class imbalance propose new algorithms that can be applied to the initial dataset as a preprocessing step to remove class imbalance. For example, in the article by Koziarski et al. [3], a new algorithm for classifying multiclass data is proposed. The authors refer to it as a multiclass Combined Cleaning and Resampling (MC-CCR) algorithm Koziarski et al. [3]. The algorithm includes a cleaning stage aimed at reducing the impact of overlapping class distributions on the effectiveness of trained algorithms. MC-CCR is less affected by the loss of information regarding interclass relationships compared to traditional multiclass decomposition strategies. Based on the results of experimental studies conducted on numerous multiclass imbalanced benchmark datasets, the proposed approach demonstrated high robustness to noise, as well as superior quality compared to state-of-the-art methods.

However, the class imbalance is rarely solved without preprocessing the dataset according to Rezvani et al. [11]. Academic literature rarely offers a solution by keeping the class imbalance and adjusting the model using its parameters in a way that the model is not affected by class imbalance. In this paper, this type of solution is discussed. We propose an innovative methodology for conducting classification analysis on multiclass public datasets (with more than two classes). In this approach, observations are modeled without additional preprocessing. Several classification models including Random Forest (RF), Support Vector Machines (SVM), and Decision Tree (DT) with appropriately selected parameters are utilized with the methodology. The parameters of these models and the KFold function, which prepares the dataset for classification analysis, are specific to each set of observations. The application of the methodology constructs effective classification algorithms for all analyzed datasets, achieving parameter values that surpass those of the model MC-CCR [3] and other authors. The conducted comparative analysis shows that the methodology achieves high values for the evaluation metrics on the 6 tested datasets (see section Results). In most cases, the results of the methodology exceed the corresponding results of other researchers. Our methodology is distinguished from other similar ones with its simple organization, speed and efficiency. These advantages of the novel methodology make it particularly suitable for cases when additional model complexity is not necessary. The next section describes the models, while the section Results discusses our results.

2. The Methodology

The proposed methodology is generalized, and therefore, it can be used with various classification methods. All experiments are conducted in Python 3.7., the Spyder environment. The proposed methodology extends the standard classification approach by first, introducing random shuffling prior to splitting the dataset into training and test subsets as it is in the standard procedure. With this step, we explore the effectiveness of other techniques for avoiding over- and underfitting in the case of class imbalance. Second, the focus of the methodology is on tuning the parameters of the model so that class imbalance might not affect the outcome of the model. The adjustment made is based on empirical experiments and aims to show that efficient model performance can be achieved also by finding the appropriate values of the model. This is a novel way to find a simple model that is appropriate for data. Although parameter tuning is not a new concept, its benefits have not been explored much. Therefore, our methodology fills this gap by showing that finding the appropriate parameters can be a simple and effective solution to the class imbalance issue in multiclass classification. Combining those new steps results in a methodology that is easy and simple to apply with various classification models.

2.1. Methodology Overview

Figure 1 demonstrates the proposed methodology.

DOI: https://doi.org/10.54216/FPA.180215
Received: July 14, 2024 Revised: October 21, 2024 Accepted: January 06, 2025

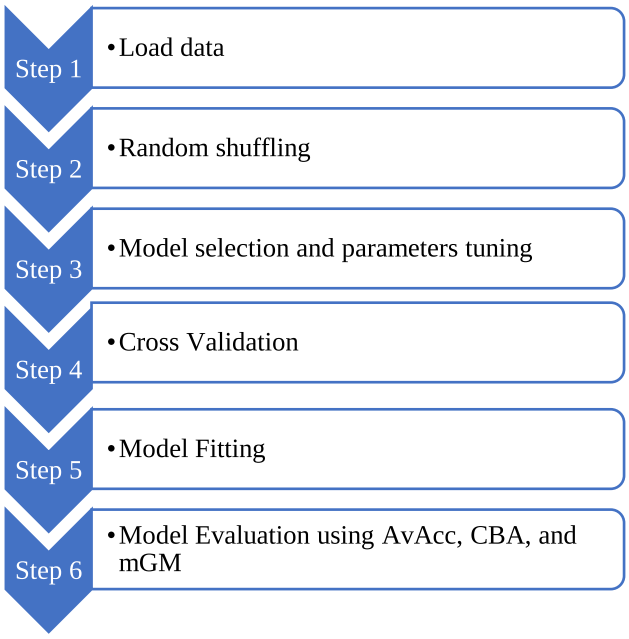

Figure 1: Summary of the Proposed Methodology
Source: authors

Step 1. Load the dataset. Define X variables that are used as predictors. Unlike the standard methodology that requires standardization of the X variables, we do not utilize this transformation. The X variables are used without any pre-processing. Then, define the Y variable, which is a categorical target variable. As the paper is aimed at multiclass classification, the target variable contains more than two categories. Each category is called a class. Therefore, this is called multiclass classification.

Step 2. Random shuffling of the data:

```
np.random.seed (28)
permuted_indices = np.random.permutation(len(Y))
```

This step rearranges the initial dataset randomly. This step is necessary due to the class imbalance issue. Class imbalance in multiclass problems results in uneven distribution of observations of each class so that one or several classes may prevail over the rest. Class prevalence causes biased estimation towards the prevailing class [4]. Prediction of minority class becomes inaccurate. The issue is deepened in the case of multiclass classification, as each class would be affected biased prediction depending on its minority/majority [4]. In balanced classification, the second step is usually a train/test split to avoid under and overfitting [4]. In imbalance classification, however, train/test split is not enough to handle the bias from class imbalance [4]. Another approach is necessary. Therefore, we examine whether the random shuffling of the whole dataset can be an effective way to predict all classes correctly if applied as a prior step to cross validation.

Step 3. Selection of a classification model and parameters. In this step, several machine learning (ML) algorithms are applied to each dataset (RF, SVM, and DT). Their performance is compared and analysed in the Results section. In this step, the machine learning algorithms used can be extended outside the three models we test. Depending on the data, another classification model can be better suited; therefore, it can be applied to replace the RF, SVM or DT models we use. However, the parameters of every classification model should be adjusted to describe the data best. Regardless of the ML model used, parameters tuning is a necessary step as each parameter aims at adjusting the functional form of the model to the data characteristics. The process of finding the appropriate parameters to handle class imbalance demonstrates the research value of this paper. We demonstrate that parameters tuning can be a key step in simplifying the model in multiclass classification given the class imbalance.

We use three ML models to perform step 3. These are the random forest (RF), support vector machines (SVM) and the decision tree classifier (DT). Every ML model has its own parameters. To tune them, start with the most appropriate parameters values based on Python's documentation. Based on classification metrics, evaluate the model performance and readjust the parameters until the output is satisfactory. It is important to note that many well-performing models may be detected based on tuning parameters. This leads to the detection of many possible

DOI: https://doi.org/10.54216/FPA.180215
Received: July 14, 2024 Revised: October 21, 2024 Accepted: January 06, 2025

well-performing models among which the researcher can select the one that reflects the data characteristics best. In this research, we present some of the parameters of the RF, SVM and DT that resulted in well-performing models according to the criteria set in Step 6 and the proposed methodology. In our cases, the parameters that are adjusted are:

3.1 Random Forest (RF) is a machine-learning algorithm that combines many decision trees to create a set of trees, so to create a forest. Each tree makes a classification or a choice. The forest chooses the classification with most of the choices [10] and Ali et al. [13].

The Python function used in these experiments is RandomForestClassifier(n_estimators ,max_depth ,max_features, class_weight='balanced', random_state=..).

In all experiments with the random forest, we keep class_weight='balanced' to account for the class imbalance. Also, experiments are conducted using different values for n_estimators , max_depth and random_state. The parameter 'n_estimators' defines the number of trees used in the model, the parameter 'max_depth' defines the number of nodes, and 'random_state' controls the reproducibility of the model. 'Max_features' defines the number of features to use when looking for the best split of the tree. Each value for 'random_state' defines the set of random values in which the experiments will be conducted. It guarantees that the same results will be produced when the same set of random values is used. We have presented the parameters that predicted all classes well in Table 3 of the next section.

3.2 Support Vector Machines (SVM) model defines a hyperplane that splits the observations of different classes (two or more). The hyperplane is the decision boundary, it is a separator between classes. The full description of the Support Vector Machines model can be found in James et al. [4], Wien et al. [14], Ke et al. [15], and many other sources in the literature [16]. The function we use is SVC(C=1, kernel='rbf', gamma=,class_weight='balanced'). In the SVM model the parameters 'class_weight' and 'kernel' are kept to 'balanced' and 'rbf' in all experiments. The parameter class_weight='balanced' is used to handle the class imbalance in the data. 'Kernel' defines the function for getting the hyperplane. The default value is 'rbf'. However, the parameters C and Gamma are tested with different values. The parameter 'gamma' defines the value for the coefficient of the kernel function. Parameter C is a regularization parameter, whose default value in Python is 1. It controls the trade-off between accuracy and variance. We have presented the parameters that predicted all classes well in Table 4 of the next section.

3.3 Decision tree classifier (DT) is a machine learning model used mainly for classification analysis. Each tree consists of nodes and branches. Each node represents features in a group that is to be classified, and each branch achieves a value that the node can take, see for example papers [8,9,13]. Instances are classified starting at the root node and sorted based on their feature values. The function used is DecisionTreeClassifier(max_depth, class_weight='balanced', random_state=). Like the rest of the models, the parameter class_weight is fixed to 'balanced' to account for the class imbalance. The parameters 'max_depth' and 'random_state' are adjusted. They have the same meaning as in the random forest model. We have presented the parameters that predicted all classes well in Table 5 of the next section.

Step 4. Splitting the dataset into training and testing subsets using the KFold command. There are two types of cross validation in class imbalance issue. The first is Kfold and the second is stratified k-fold cross validation. Although stratified cross validation can be more suitable, empirical data has shown that the two types of cross validation work well in imbalanced datasets. Therefore, we performed experiments only with kfold cross validation. The kfold cross validation function is applied with 4 splits and random shuffling. The parameter 'shuffle' is set to true, as the aim is to mix imbalanced classes to have training and test samples that are highly diverse. The number of splits is set to 4 but it can be set to any other value different from 1. The number of splits depends on the researcher's experience and data. However, experiments with a different value of random_state have been conducted. KFold(n_splits=4, shuffle=True,random_state=..). We selected the value of random_state that produced the best classification metrics for each ML model. As Table 3 in the Results section shows the random_state value in the cross-validation, function is different for each ML model.

Step 5. Fitting of the model. After randomly shuffling the data and splitting the dataset into training and test sets using the cross-validation function in Step 4, we fit each of the classification models with the parameters shown in Step 3.

Step 6. Evaluation of the model using parameters: AvAcc, CBA, and mGM. This is a model estimation. Each classification model is evaluated by computing a special matrix called the confusion matrix. If we denote this

DOI: https://doi.org/10.54216/FPA.180215
Received: July 14, 2024 Revised: October 21, 2024 Accepted: January 06, 2025

matrix as C = (c_{ij}), where c_{ij} are the elements of the matrix, and m represents the number of classes. The meaning of each element c_{ij} is as follows: it is the number of observations from class i in the dataset that the model predicts as belonging to class j. The elements c_{ii}, i = 1,..., m represent the number of correctly predicted observations, while the elements c_{ij}, i,j=1,...,m, i<>j represents the misclassified observations or errors of the model.

Following the authors Branco et al. [2], Koziarski et al. [3], three parameters are defined: Average Accuracy (AvAcc), Class Balance Accuracy (CBA), and multi-class G-measure (mGM). They are used for evaluating the classification models:

$$AvAcc(Accuracy) = \frac{\sum_{i=1}^{m} c_{ii}}{\sum_{i,j=1}^{m} c_{ij}}, \quad (1)$$

$$\text{CBA} = \frac{1}{m}\sum_{i=1}^{m} \frac{c_{ii}}{\max\left(\sum_{j=1}^{m} c_{ij}, \sum_{j=1}^{m} c_{ji}\right)}, \quad (2)$$

$$mGM = \sqrt[m]{\prod_{i=1}^{m} \frac{c_{ii}}{\sum_{j=1}^{m} c_{ij}}}\ . \quad (3)$$

The process of tuning parameters includes running steps 1-6 with different values of the 'random_state' of the kfold cross validation function and various values of the parameters of the ML models. Based on the AvACC, GBA and mGM, the values of the parameters are changed, and steps 1-6 are run again. We run this methodology until we get satisfying results for AvAcc, GBA, mGM and confusion matrix. The tables in the next sections demonstrate all parameters we selected using steps 1-6 to perform correct multiclass prediction and solve the class imbalance issue. The tables demonstrate only the efficient models. An efficient model is the one resulting in the best classification scores and predicts all classes correctly (as defined in step 6). We compare our results with other authors. Our experiments illustrate an important finding: Find an efficient' model for each ML group does not require hundreds of reruns of steps 1-6. The loop stops when a model good enough according to the presented criteria is identified. During our experiments, an efficient model could be found until the tenth rerun. This finding confirms that parameters tuning may not be a complex and time-consuming step, stressing its underestimated benefits.

2.2. Data Overview

The suggested methodology is utilized in a set of observations, presented in Tables 1 and 2. Table 1 describes the six datasets used, the number of observations, variables and classes in the target variable. It shows that the number of variables is relatively small, so feature selection is not necessary at this point. All datasets present the multiclass problem as the target variable contains more than two classes. In addition, the methodology is tested on smaller and larger datasets.

Table 1: Datasets

Name	Samples	Variables	Classes	Class Distribution
Balance	625	4	3	288/49/288
Contraceptive	1473	9	3	629/333/511
Hayes-Roth	160	4	3	65/64/31
New-thyroid	215	5	3	150/35/30
Page-blocks	5472	10	5	4913/329/28/87/115
Thyroid	7200	21	3	166/368/6666

The column "Class Distribution" demonstrates how the issue of class imbalance is represented in each of the datasets. For instance, the target variable in the dataset 'Balance' has 288 observations in the first class, 49 in the second, and 288 in the third class. The same structure applies to the rest of the datasets. As the last column shows, heavy class imbalance is present in each dataset. This issue becomes more complicated as it relates to multiclass labels. Table 2 presents the sources of the datasets.

DOI: https://doi.org/10.54216/FPA.180215
Received: July 14, 2024 Revised: October 21, 2024 Accepted: January 06, 2025

Table 2: Public access to the datasets

Name	Sources
Balance	https://archive.ics.uci.edu/dataset/12/balance+scale
Contraceptive	https://archive.ics.uci.edu/dataset/30/contraceptive+method+choice
Hayes-Roth	https://archive.ics.uci.edu/dataset/44/hayes+roth
New-thyroid	https://sci2s.ugr.es/keel/dataset.php?cod=66
Page-blocks	http://archive.ics.uci.edu/dataset/78/page+blocks+classification
Thyroid	https://networkrepository.com/thyroid-disease-thyroid0387.php
	https://archive.ics.uci.edu/dataset/102/thyroid+disease

Several classification models, namely RF, SVM, and DT, with appropriately selected parameters are utilized for conducting the classification analysis on these datasets. Results from the experiments conducted with this methodology are shown in the next section.

3. Results

This section describes the final values of the parameters in every efficient ML model we ran (Tables 3-5). In addition, we compare our results to other authors. For example, Koziarski et al. [3] apply their algorithm to 20 public datasets, which are described in Table 1[3]. We apply our methodology to four of the datasets used in [3]. These are Contraceptive, Hayes-Roth, New-thyroid, and Page-blocks. The results of the classification analysis conducted are described in Tables 3, 4, and 6, respectively for each model. The parameters with which the models were applied are added to the tables. The computations were conducted on a laptop with 1.50 GHz Intel(R) Core(TM) and 8 GB RAM, running on Windows with Python 3.7 in the Anaconda environment.

Table 3 shows the final parameters of the Random Forest model we selected. The output demonstrates that the best results in terms of AvAcc, CBA and mGM on each dataset are achieved using different parameters of the Random Forest model for each dataset. Therefore, Table 3 demonstrates that parameter adjustment may be better conducted on each dataset rather than applying the same parameters on all datasets. The reason for that is that the parameters should best describe the characteristics of the data. Parameters adjustment reflects data anomalies. Parameters aim at capturing data anomalies by adjusting the equation of the classification model and making it flexible. Every model's parameter reflects an equivalent parameter in the equation of the ML algorithm. Adjusting the model parameters results in adapting the ML equation to the data specifics. Therefore, using different parameters on each dataset may be the more sensible approach.

As seen in the table the same dataset may produce different metrics depending on the values of the parameters in the Random Forest function. By adjusting the parameter values in the random forest function, the classification metrics on the same dataset can be quite different. For instance, Table 3 displays the parameters of the random forest function with which the best result was achieved on each dataset. However, throughout the experiment, each dataset was tested with various values of the RF parameters. They did not perform so well, so they are not visible in Table 3. An important finding is confirmed – an ML algorithm may be a more appropriate model than another one depending on the data. As Table 3 illustrates, classification metrics on the same dataset vary depending on the ML model used. As every ML model aims at tackling a specific data structure, it is also reasonable to perform parameters tuning within each ML model.

3.1. Random Forest, Support Vector Machines, and Decision Tree Classifier

As Table 3 shows, this research proposes an algorithm that is more effective than applying pre-processing techniques for class imbalance like SMOTE. Measures like AvAcc, CBA and mGM are higher because of the parameter adjustment proposed in this research. As Table 3 shows, parameter adjustment does not provide a single model that is efficient. Instead, it provides many models that have better performance than algorithms with pre-processing techniques for class imbalance. The proposed approach reduces the complexity of the model and allows for easier interpretation. It also saves time by reducing the time for building, fitting, and selecting the model.

DOI: https://doi.org/10.54216/FPA.180215
Received: July 14, 2024 Revised: October 21, 2024 Accepted: January 06, 2025

Table 3: Comparison of the Proposed Methodology with Random Forest Model to Koziarski [3]

Name	Indicator	Maximum values (Koziarski *et al.* [3])	Random Forest Parameters
Balance			RandomForestClassifier(n_estimators=120, max_depth=3, class_weight='balanced', random_state=6) KFold(n_splits=4, shuffle=True, random_state=167)
	AvAcc [%]	82.87	89.17
	CBA [%]	64.92	70.34
	mGM [%]	62.22	74.65
Contraceptive			RandomForestClassifier(n_estimators=100, max_depth=9, random_state=422) KFold(n_splits=4, shuffle=True,random_state=667)
	AvAcc [%]	55.09	62.5
	CBA [%]	52.97	60.55
	mGM [%]	52.60	60.38
Hayes-Roth			RandomForestClassifier(n_estimators=60, max_depth=9, random_state=422) KFold(n_splits=4, shuffle=True, random_state=561)
	AvAcc [%]	92.11	95.0
	CBA [%]	90.03	92.06
	mGM [%]	85.16	94.99
New-thyroid			RandomForestClassifier(n_estimators=50, max_features="auto") KFold(n_splits=4, shuffle=True, random_state=500)
	AvAcc [%]	96.18	100.0
	CBA [%]	95.22	100.0
	mGM [%]	93.46	100.0
Page-blocks			RandomForestClassifier(n_estimators=80, max_features="auto", random_state=6) KFold(n_splits=4, shuffle=True, random_state=867)
	AvAcc [%]	83.71	98.0
	CBA [%]	84.63	84.27
	mGM [%]	82.55	90.45

DOI: https://doi.org/10.54216/FPA.180215
Received: July 14, 2024 Revised: October 21, 2024 Accepted: January 06, 2025

Thyroid			RandomForestClassifier(n_estimators=100, max_depth=9, random_state=42) KFold(n_splits=4, shuffle=True,random_state=519)
	AvAcc [%]	85.34	99.89
	CBA [%]	81.99	98.79
	mGM [%]	81.86	99.96

Table 4 displays the parameters that produce the best results on each dataset using support vector machines. Table 5 does the same for the Decision Tree Classifier. All tables provide the same insights as Table 3. In this research, the criteria for selecting the best parameters for each dataset are high enough classification metrics and good enough confusion metrics. As the tables show, the proposed methodology can produce many models that are a good fit by adjusting the parameters.

Table 4: Comparison of the Proposed Methodology with SVM (kernel='rbf') to Koziarski [3]

Name	Indicator	Maximum values (Koziarski *et al.* [3])	SVM (kernel='rbf') Parameters
Balance			SVC(C=100, kernel='rbf', gamma='auto', class_weight='balanced') KFold(n_splits=4, shuffle=True,random_state=684)
	AvAcc [%]	82.87	96.79
	CBA [%]	64.92	86.68
	mGM [%]	62.22	97.78
Contraceptive			SVC(C=0.1, kernel='rbf', gamma='auto', class_weight='balanced') KFold(n_splits=4, shuffle=True,random_state=16)
	AvAcc [%]	55.09	60.0
	CBA [%]	52.97	53.47
	mGM [%]	52.60	57.08
Hayes-Roth			SVC (C=100, kernel='rbf', gamma='auto', class_weight='balanced') KFold(n_splits=4, shuffle=True,random_state=469)
	AvAcc [%]	92.11	95.0
	CBA [%]	90.03	95.48
	mGM [%]	85.16	95.54
New-thyroid			SVC (C=1, kernel='rbf', gamma=0.01, class_weight='balanced')

DOI: https://doi.org/10.54216/FPA.180215
Received: July 14, 2024 Revised: October 21, 2024 Accepted: January 06, 2025

			KFold(n_splits=4, shuffle=True,random_state=3)
	AvAcc [%]	96.18	98.14
	CBA [%]	95.22	95.4
	mGM [%]	93.46	96.15
Page-blocks			SVC (C=1000, kernel='rbf', gamma=0.000001, class_weight='balanced') KFold(n_splits=4, shuffle=True, random_state=147)
	AvAcc [%]	83.71	92.
	CBA [%]	84.63	61.8
	mGM [%]	82.55	81
Thyroid			SVC(C=1000, kernel='rbf', gamma=10, class_weight='balanced') KFold(n_splits=4, shuffle=True,random_state=293)
	AvAcc [%]	85.34	96.78
	CBA [%]	81.99	82.16
	mGM [%]	81.86	88.12

Table 4 shows that the proposed methodology can outperform existing research Koziarski et al. [3] also when using the support vector machines. In some cases, like the 'Balance' dataset, the performance improvement is significant. This is also valid for the decision tree classifier, e.g. the performance on the thyroid dataset is better using our methodology (see Table 5).

Table 5: Comparison of the Proposed Methodology with Decision Tree Model to Koziarski [3]

Name	Indicator	Maximum values (Koziarski *et al.* [3])	Decision Tree Parameters
Balance			DecisionTreeClassifier(max_depth=8, class_weight='balanced', random_state=139) KFold(n_splits=4, shuffle=True,random_state=139)
	AvAcc [%]	82.87	84.71
	CBA [%]	64.92	66.22
	mGM [%]	62.22	65.65
Contraceptive			DecisionTreeClassifier(max_depth=6, class_weight='balanced', random_state=97) KFold(n_splits=4, shuffle=True, random_state=45)
	AvAcc [%]	55.09	58.42
	CBA [%]	52.97	52.79

DOI: https://doi.org/10.54216/FPA.180215
Received: July 14, 2024 Revised: October 21, 2024 Accepted: January 06, 2025

	mGM [%]	52.60	59.13
Hayes-Roth			DecisionTreeClassifier(max_depth=6, class_weight='balanced', random_state=97) KFold(n_splits=4, shuffle=True, random_state=429)
	AvAcc [%]	92.11	97.5
	CBA [%]	90.03	95.56
	mGM [%]	85.16	97.14
New-thyroid			DecisionTreeClassifier(max_depth=6, class_weight='balanced', random_state=97) KFold(n_splits=4, shuffle=True,random_state=749)
	AvAcc [%]	96.18	98.11
	CBA [%]	95.22	96.30
	mGM [%]	93.46	99.07
Page-blocks			DecisionTreeClassifier(max_depth=14, class_weight='balanced', random_state=98) KFold(n_splits=4, shuffle=True,random_state= 817)
	AvAcc [%]	83.71	97.00
	CBA [%]	84.63	84.15
	mGM [%]	82.55	90.71
Thyroid			DecisionTreeClassifier(max_depth=6, class_weight='balanced', random_state=97) KFold(n_splits=4, shuffle=True,random_state=45)
	AvAcc [%]	85.34	99.67
	CBA [%]	81.99	97.66
	mGM [%]	81.86	99.54

As Tables 3-5 illustrate, tuning the parameters in every model resulted in an efficient model that can predict well all classes despite the class imbalance and outperform other algorithms focused on modifying the model rather than parameters tuning. However, parameters tuning involves observing several criteria before deciding whether the model is a good fit. These are AvAcc, CBA, mGM and confusion matrices. Observing confusion matrices is a key step in the model evaluation. Confusion matrices reveal how accurately each class is predicted. Cases when AvAcc, CBA and mGM are high but the confusion matrix demonstrates that some classes are not predicted accurately cannot be defined as a good fit. In this case, the confusion matrix can highlight a biased model. In this case, parameters tuning continues until a model where AvAcc, CBA and mGM are high enough and the confusion matrix confirms that all classes are predicted accurately. If these criteria are fulfilled, tuning the parameters stops. The model with these parameters is selected. Our experiments showed that finding parameters that fulfil these criteria often happens during the first several trials if the initial parameters values were set in accordance with data characteristics and the appropriate values based on the Python documentation. Therefore, the adjustment of parameters does not start with random values. This approach to parameter tuning often saves time and becomes efficient as the good fit can be captured quickly. Another important consideration is that parameters tuning suggest

DOI: https://doi.org/10.54216/FPA.180215
Received: July 14, 2024 Revised: October 21, 2024 Accepted: January 06, 2025

that more than one good fit is available. When the researchers stop depends on their aim. In some cases, discovering the first good fit may be enough. In other cases, finding several good fits would be good. Every good fit would have different parameters, but all good fits should have AvAcc, CBA and mGM that are high enough and the confusion matrix confirms that all classes are predicted accurately.

Another important finding is that random resampling prior to cross validation can be an effective step to avoid under and overfitting when class imbalance is present. Tables 3-5 demonstrate that our results outperform other authors. Parameters tuning contributes to this outcome. However, handling class imbalance also has an important contribution to this outcome. Random shuffling prior to cross validation allows more class diversity in training and test samples, which mitigates the effects of class imbalance prior to parameters tuning. Most Python classification algorithms have a built-in parameter called 'class_weight' that is set to 'balanced' to tackle class imbalance. However, in multiclass problems with heavy class imbalance, using this parameter is not enough to reduce the bias from the imbalance. Therefore, the bias from class imbalance cannot be handled by adjusting the remaining parameters. Therefore, the class imbalance should be handled separately and then adjustment of the parameters is made. In this paper, we illustrate that random sampling prior to cross validation can mitigate the bias coming from class imbalance, which also helps to identify model's parameters that would result in accurate prediction of all classes.

3.2. Comparison with other authors

The construction of algorithms for classification analysis of big multi-class data has attracted the attention of many researchers, who propose diverse and efficient algorithms of high complexity, such as those proposed in the following articles [15-22]. In this section, we compare our results presented in section 3.1 with those of other authors in their papers 19-22]. The experimental results from MC-NRO [19] are displayed in Table 6. Compared with the corresponding measures computed by our methodology with the Random Forest model (see the second column of Table 6) we note that the values are higher than the MC-NRO values. Our Random Forest model outperforms the MC-NRO method.

Table 6: Results after applying the MC-NRO method introduced by Shen et al. [19]. Compared to our RF results

Name	Indicator	Random Forest Model	MC-NRO Tables 7,8,9 [19]
Balance			
	AvAcc [%]	89.17	85.70
	CBA [%]	70.34	69.53
	mGM [%]	74.65	64.39
Contraceptive			
	AvAcc [%]	62.5	54.01
	CBA [%]	60.55	52.95
	mGM [%]	60.38	52.85
Hayes-Roth			
	AvAcc [%]	95.0	93.64
	CBA [%]	92.06	92.45
	mGM [%]	94.99	88.12

DOI: https://doi.org/10.54216/FPA.180215
Received: July 14, 2024 Revised: October 21, 2024 Accepted: January 06, 2025

New-thyroid			
	AvAcc [%]	100.0	97.89
	CBA [%]	100.0	95.33
	mGM [%]	100.0	97.96
Page-blocks			
	AvAcc [%]	98.0	88.57
	CBA [%]	84.27	87.43
	mGM [%]	90.45	88.37
Thyroid			
	AvAcc [%]	99.89	96.67
	CBA [%]	98.79	91.93
	mGM [%]	99.96	93.10

Results obtained with the MC-MBRC algorithm [20] are presented in Table 7 for some datasets (see Table 7). Our RF results are introduced as a column in Table 7. The Random Forest model showed higher results received for measures AvAcc and CBA, which indicates the model's efficiency for multiclass datasets.

Table 7: Results after applying the MC-MBRC method introduced by Ma et al. [20]. Compared to our RF results

Name	Indicator	Random Forest Model	Algorithm MC-MBRC (Tables 3 and 5, Ma *et al.* [20])
Contraceptive			
	AvAcc [%]	62.5	60.53
	CBA [%]	60.55	33.13
New-thyroid			
	AvAcc [%]	100.0	95.36
	CBA [%]	100.0	85.45
Page-blocks			
	AvAcc [%]	98.0	96.87
	CBA [%]	84.27	57.27
Thyroid			
	AvAcc [%]	99.89	94.36

DOI: https://doi.org/10.54216/FPA.180215
Received: July 14, 2024 Revised: October 21, 2024 Accepted: January 06, 2025

CBA [%]	98.79	63.64

We compare results served by Ongko et al. [21] for datasets Balance и Newthyroid. The value of Av Acc for the Balance dataset is 86.50% and for the Newthyroid dataset is 97.2% (Table 4 [21]). The value of mGM for the Balance dataset is 91.50% and for the Newthyroid dataset is 89.80% (Table 5 [21]). Our best results for these datasets are for the Balance dataset. The SVM model and the Newthyroid dataset 96.79% and 97.78% obtain the metrics Av Acc and mGM. The RF model 100% and 100% obtains the same measures. Thus, our methodology increases the values of Ac Acc and mGM for these datasets.

Table 8 describes some of the datasets and results obtained by Grina et al. [22]. The measurements used by Grina et al. [22] are mGM and F1-score. Comparing Grina's results in Table 8 with our findings in Tables 3 and 4, we conclude that our values compete with Grina's results.

Table 8: Results after applying the Grina's methodology [22]. Our results from Tables 4 and 5 are given in brackets. Compared to our RF results.

Name	Indicator	SVM model Table 2 [22]	DT model Table 4 [22]	Our RF model
Balance				
	mGM [%]	88.5 (97.78)	89.7 (65.65)	74.65
Contraceptive				
	mGM [%]	54.7 (57.08)	95.2 (59.13)	60.38
New-thyroid				
	mGM [%]	96.6 (96.15)	53.8 (99.07)	100.0
Page-blocks				
	mGM [%]	59.2 (81.0)	55.8 (90.71)	90.45
Thyroid				
	mGM [%]	98.7 (88.12)	99.0 (99.54)	99.96

The last Tables 6, 7, and 8 present the results of other authors on multiclass set classification problems. These results are comparable and even lower than the results of the proposed methodology.

4. Conclusion

These findings lead to several important discoveries for handling class imbalance in multiclass problems. First, parameter tuning of ML models can be an effective strategy to handle class imbalance. However, the selection of initial parameter values should be made based on the dataset characteristics and Python documentation. If that is the case, a good fit can be found in the initial rounds of tuning. Second, random shuffling prior to cross validation can be a key to resolving the bias coming from multiclass imbalance. Another important finding is that the best results can be achieved when random shuffling, cross validation and parameter tuning are combined. We demonstrated this in the proposed methodology. In addition, parameters tuning should be done for each dataset and for each ML algorithm used. As the experiments show tuning the parameters of classification models in case of class imbalance may be an effective way to resolve the bias that would come with class imbalance. In some cases, this approach may be better than using pre-processing techniques to handle the issue with class imbalance. The advantages of the proposed algorithm lie in the simplicity and fast resolving of the class imbalance issue, which makes this methodology appropriate as a data exploratory technique or a full classification methodology

DOI: https://doi.org/10.54216/FPA.180215
Received: July 14, 2024 Revised: October 21, 2024 Accepted: January 06, 2025

even in multiclass datasets. However, tuning the parameters requires experiments and evaluation of the effects of the given parameter on the model. Using cross validation is a way to avoid bias in classification in this approach. However, more research should be conducted on the parameters influence on the model's performance in multiclass datasets. Therefore, we recommend the proposed algorithm to be used to explore and understand the underlying data better. It can be used as a classification technique in the case of multiclass issue, when the aim is classification. We do not recommend using this approach in multiclass problems for feature selection as parameter's effects on the selection of variables is not addressed in this research.

Funding: "This research received no external funding"

Conflicts of Interest: "The authors declare no conflict of interest."

References

[1] J. Manyika, M. Chui, B. Brown, and J. Bughin, "Big Data: The Next Frontier for Innovation, Competition," 2011. [Online]. Available: https://www.semanticscholar.org/paper/Big-data%3A-The-next-frontier-for-innovation%2C-and-Manyika/91b63db746becca15090963a8990dfe2b5103799. [Accessed: Oct. 28, 2024].

[2] P. Branco, L. Torgo, and R. P. Ribeiro, "Relevance-based evaluation metrics for multi-class imbalanced domains," in *Advances in Knowledge Discovery and Data Mining - 21st Pacific-Asia Conference, PAKDD 2017*, Jeju, South Korea, May 23–26, 2017, Part I, 2017, pp. 698–710.

[3] M. Koziarski, M. Woźniak, and B. Krawczyk, "Combined Cleaning and Resampling algorithm for multi-class imbalanced data with label noise," *Knowledge-Based Systems*, vol. 204, p. 106223, 2020.

[4] J. G. James, D. Witten, T. Hastie, and R. Tibshirani, *An Introduction to Statistical Learning: with Applications in R*, 2nd ed., Springer, 2021. [Online]. Available: https://www.statlearning.com/. [Accessed: Nov. 20, 2024].

[5] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: synthetic minority over-sampling technique," *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, 2002.

[6] H. Han, W.-Y. Wang and B.-H. Mao, "Borderline-SMOTE: a new over-sampling method in imbalanced data sets learning," in *Advances in Intelligent Computing. ICIC 2005*, D. S. Huang, X. P. Zhang, and G. B. Huang, Eds., Berlin, Heidelberg: Springer, 2005, vol. 3644, pp. 878–887.

[7] S. Maldonado, C. Vairetti, A. Fernandez, and F. Herrera, "FW-SMOTE: A feature-weighted oversampling approach for imbalanced classification," *Pattern Recognition*, vol. 124, p. 108511, 2022.

[8] A. Dey, "Machine learning algorithms: a review," *International Journal of Computer Science and Information Technologies*, vol. 7, no. 3, pp. 1174–1179, 2016.

[9] S. B. Kotsiantis, "Supervised Machine Learning: A Review of Classification Techniques," *Informatica*, vol. 31, pp. 249–268, 2007.

[10] R. Hansch, *Handbook of Random Forests: Theory and Applications for Remote Sensing*, Series in Computer Vision, World Scientific, 2021.

[11] S. Rezvani and X. Wang, "A broad review on class imbalance learning techniques," *Applied Soft Computing*, vol. 143, 2023.

[12] E. S. Agung, A. P. Rifai, and T. Wijayanto, "Image-based facial emotion recognition using convolutional neural network on Emognition dataset," *Sci Rep*, vol. 14, p. 14429, 2024.

[13] J. Ali, R. Khan, N. Ahmad, and I. Maqsood, "Random Forests and Decision Trees," *International Journal of Computer Science*, vol. 9, no. 5, pp. 1694–0814, 2012.

[14] M. Wien, H. Schwarz, and T. Oelbaum, "Performance analysis of SVC," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 17, pp. 1194–1203, 2007.

[15] T. Ke, X. Ge, F. Yin, L. Zhang, Y. Zheng, C. Zhang, J. Li, B. Wang, and W. Wang, "A general maximal margin hyper-sphere SVM for multi-class classification," *Expert Systems with Applications*, vol. 237, p. 121647, 2024.

[16] S. Sridhar and S. Anusuya, "A dual algorithmic approach to deal with multiclass imbalanced classification problems," *Big Data Research*, vol. 38, p. 100484, 2024.

DOI: https://doi.org/10.54216/FPA.180215
Received: July 14, 2024 Revised: October 21, 2024 Accepted: January 06, 2025

[17] C.-F. Tsai, K.-C. Chen and W.-C. Lin, "Feature selection and its combination with data over-sampling for multi-class imbalanced datasets," *Applied Soft Computing*, vol. 153, p. 111267, 2024.

[18] Q. Dai, J.-W. Liu, and Y. Liu, "Multi-granularity relabeled under-sampling algorithm for imbalanced data," *Applied Soft Computing*, vol. 124, p. 109083, 2022.

[19] S. Shen, Z. Li, Z. Huan, F. Shang, Y. Wang, and Y. Chen, "Neighborhood repartition-based oversampling algorithm for multiclass imbalanced data with label noise," *Neurocomputing*, vol. 600, p. 128090, 2024.

[20] T. Ma, S. Lu, and C. Jiang, "A membership-based resampling and cleaning algorithm for multi-class imbalanced overlapping data," *Expert Systems with Applications*, vol. 240, p. 122565, 2024.

[21] E. Ongko and H. Hartono, "Hybrid approach redefinition-multi class with resampling and feature selection for multi-class imbalance with overlapping and noise," *Bulletin of Electrical Engineering and Informatics*, vol. 10, no. 3, pp. 1718–1728, 2021.

[22] F. Grina, Z. Elouedi, and E. Lefevre, "Re-sampling of multi-class imbalanced data using belief function theory and ensemble learning," *International Journal of Approximate Reasoning*, vol. 156, pp. 1–15, 2023.

DOI: https://doi.org/10.54216/FPA.180215
Received: July 14, 2024 Revised: October 21, 2024 Accepted: January 06, 2025