

A Performance Analysis of Classifiers on Imbalanced Data

Nathan F. Garcia^a, Rômulo A. Strzoda^b, Giancarlo Lucca^c and Eduardo N. Borges^d

Centro de Ciências Computacionais, Universidade Federal do Rio Grande, Rio Grande, Brazil

Keywords: Machine Learning, Data Imbalance, Supervised Classification.

Abstract: In the machine learning field, there are many classification algorithms. Each algorithm performs better in certain scenarios, which are very difficult to define. There is also the concept of grouping multiple classifiers, known as ensembles, which aim to increase the model generalization capacity. Comparing multiple models is costly, as, for certain cases, training classifiers can take a long time. In the literature, many aspects of the data have already been studied to help in the task of classifier selection, such as measures of diversity among classifiers that form an ensemble, data complexity measures, among others. In this context, the main objective of this work is to analyze class imbalance and how this measure can be used to guide the selection of classifiers. We also compare the model's performances when using class balancing techniques such as oversampling and undersampling.

1 INTRODUCTION

Machine Learning (ML) (Bishop, 2006) is a field of study in the field of artificial intelligence. Among others, ML is used to deal with classification problems (Hart et al., 2000). From a supervised point of view, such a problem consists of finding a model or a function that can identify patterns and describe different classes of data. Therefore, the purpose of classification is to label new examples by applying the model or learned function. This model is based on a set of features extracted from available data.

There are several techniques proposed in the literature. Support Vector Machines (SVM) (Steinwart and Christmann, 2008), Decision Trees (DT) (Quinlan, 1986), Artificial neural networks (ANNs) (Haykin and Network, 2004), Fuzzy Rules Based Classification Systems (Ishibuchi et al., 2004) are examples of well-known classification algorithms. Each approach is more suitable for a specific classification problem, that is, one classification algorithm may not effectively and/or efficiently recognize some patterns in complex datasets, while another may perform optimally for the same task.

To obtain better performance, classification ensembles were proposed, combining several classifiers


to improve the model's generalization and, consequently, to develop more efficient models for identifying data classes. Several types of ensemble (Opitz and Maclin, 1999a) generation can be generalized in techniques such as Bagging, Boosting and Stacking (Quinlan et al., 1996), among others.


Selecting a specific solution to deal with a classification problem is a complicated task, considering a large number of algorithms and proposed techniques available in the literature. Each problem can have its ideal classification algorithm. Running them all and adjusting their hyperparameters is not a good idea, as the amount of time and resources will be prohibitive if the training data is high.


It is noteworthy that the best performance of the ensembles is not guaranteed and obtained in all types of data. In many cases, the ensemble's efforts are not justified since models are even worse than the base models in some situations. A deeper analysis of this issue is essential, given the wide use of classifiers in society, becoming more critical every year.


Researchers have analyzed the ensemble's performance from the point of view of distinct aspects: the diversity among classifiers, the complexity of the data sets, and the unbalance of data (Garcia et al., 2021). They could not see a strong relationship between the ensemble's performance and these aspects. However, in cases where there was an advantage in using ensembles, data were imbalanced above the average.

Classification models can be used in a variety

^a  <https://orcid.org/0000-0002-3149-9903>

^b  <https://orcid.org/0000-0001-7634-7236>

^c  <https://orcid.org/0000-0002-3776-0260>

^d  <https://orcid.org/0000-0003-1595-7676>

of scenarios, such as bankruptcy prediction (Barboza et al., 2017), discovering diseases such as cancer (Kourou et al., 2015), and flood prediction (Mosavi et al., 2018). Given the significant difficulty in selecting an algorithm based on data or classifier characteristics and the lack of studies that address this issue, the present paper aims to analyze the scenarios in which ensembles perform better than base classifiers, specifically from the perspective of imbalance. The degree of imbalance in the datasets is drastically different, making it possible to analyze behavior effectively. Some specific goals are considered: (i) analyze the performance of the best base classifiers versus the ensemble, comparing their performance using different metrics, mainly F1-Score, and (ii) compare the performance of classifiers at different degrees of imbalance, using balancing techniques such as under and oversampling.

This paper is organized as follows. The theoretical part is presented in Section 2. Section 3 introduces the related works. After that, the methodology is presented in Section 4. The obtained results are discussed in Section 5. Finally, in Section 6, the conclusions are stated.

2 THEORETICAL BACKGROUND

The development of ML techniques was based on the idea that systems can learn from data, identifying patterns that can eventually be used for decision making with low human intervention. In other words, classification refers to the scenario in which an algorithm predicts a class based on a set of labeled data. Several algorithms aim to classify specific data, whether supervised or not. Each algorithm will thrive in different scenarios, making it variable which algorithm will have the best performance for a given dataset.

2.1 Data Imbalance

A dataset is imbalanced when one class label has many more examples than another (Japkowicz and Stephen, 2002). Class imbalance is an essential factor as it is found in many fields. For instance, regarding the problem of detecting bank fraud, only a minority of transactions presents fraud. The presence of imbalance has a significant impact on the classifier performance (Japkowicz, 2000). Considering that the algorithm aims to recognize patterns to perform the classification activity, few examples of a particular class make the algorithm not learn its nuances. Furthermore, statistically, it is a tendency that the generated

model classifies a minority class as the majority class, simply due to the large distribution variation.

There are a few ways to mitigate the data imbalance problem, the most common being over and undersampling (Yap et al., 2014). The first technique, oversampling, consists of adding cases to the minority classes by replicating existing data. On the other hand, the undersampling technique eliminates cases from the majority classes. The instances are chosen randomly, and the process stops when all classes reach the same number of examples. It is important to indicate that the degree of balance is arbitrary: there is no need for equal distribution, e.g., 50% of balance considering two classes. Therefore, such techniques guarantee a degree of flexibility in experimentation.

Several problems are arising from such balancing techniques. One of them is modifying the natural distribution of events. When we balance bank transactions, we present data to the learning algorithm with a higher proportion of fraud. Furthermore, in the case of undersampling, a significant portion of cases can be lost, negatively impacting the classifier. In the case of oversampling, data duplication can generate the phenomenon known as overfitting (Hawkins, 2004), where the algorithm presents excellent results in training but bad results in real scenarios since the data is no longer representative.

2.2 Classifiers

A supervised machine learning classifier aims to recognize patterns in labeled data fitting a classification function or model. Several types of classifiers vary in many factors, such as functioning, type, and interpretability (Polat et al., 2008).

There is also the possibility of combining different classifiers, which we commonly call ensembles (Dietterich et al., 2002). They are meta-classifiers that combine multiple algorithms or classification schemas (Opitz and Maclin, 1999b). This combination makes ensemble methods have a better capacity for generalization. There are several ways to combine classifiers. In the present study, we use Stacking (Merz, 1999), Boosting (Schapire, 2013), Bagging (Ahmad et al., 2018), and Voting (Saqlain et al., 2019) techniques. In our experiments, these ensembles have used the following algorithms as base classifiers: Naive Bayes (Zhang, 2005), Logistic Regression (Dong et al., 2016), Support Vector Machines (Bhavan et al., 2019), K-Nearest Neighbors (Muliono et al., 2020), and Decision Trees (Myles et al., 2004).

Table 1: Confusion matrix example. N represents the number of instances inside the predictive model during their evaluation, it's the sum of all classification possibilities.

Actual / Predicted	Positive	Negative
Positive	TP	FN
Negative	FP	TN

$N = TP + TN + FN + FP$

2.3 Evaluation of Predictive Models

An essential step in the generation of predictive models is their evaluation. The evaluation metrics summarize the model's predictive power and provide a basis for comparison when new models are generated. It is widespread to generate several models before defining the final model. There are many hyperparameters and possible data processing tasks to be varied, and each combination generates a model to be compared. Furthermore, in the case of interpretable models, it may be found that some features do not help in predicting a particular phenomenon.

The most common metrics used in the evaluation of machine learning models use a matrix organization known as Confusion Matrix (CM) (Visa et al., 2011), which provides a summarized view of the performance of the predictive model on a case-by-case basis. That is, the possible scenarios related to a prediction are counted. The first is the case of the hit, known as True positive (TP). The opposite scenario is the other possible correct prediction, known as True Negative (TN). The other two cases are about errors: False Positives (FP) and False Negatives. Table 1 provides a more visual way to make the confusion matrix easier to understand.

Accuracy is the most common metric used in the Machine Learning field. It is the proportion of correct predictions (TP, TN) and the total instances used in evaluating the model, defined by the equation 1.

$$Accuracy = \frac{TP + TN}{N} \quad (1)$$

Precision is a predictive model evaluation metric that aims to answer the following question: What proportion of cases classified as positive is positive? The equation 2 defines the Precision.

$$Precision = \frac{TP}{TP + FP} \quad (2)$$

Recall is usually used in conjunction with Precision, as it aims to answer a question that is also relevant in the analysis of the performance of predictive models: which proportion of the positive cases was correctly classified? Equation 3 defines the Recall.

$$Recall = \frac{TP}{TP + FN} \quad (3)$$

F1-Score is the harmonic mean between Accuracy and Recall, and is defined by equation 4. When generating predictive models that use unbalanced datasets as a basis, F1 is very powerful: it is possible to summarize the model's performance between classes, with minority classes having a significant impact on the final result.

$$F1-Score = 2 * \frac{Precision * Recall}{Precision + Recall} \quad (4)$$

Area Under the ROC Curve (AUC) is an evaluation metric very good in generalizing results also in unbalanced scenarios. Obtaining them, however, is considerably more complex than the other measures. The curve that delimits its area is called the Receiver Operating Characteristic (ROC) (Hoo et al., 2017). Such a curve is constructed with the possible combinations between the proportion of True Positives and False Positives, defined through different decision thresholds.

When machine learning models are built, validation is required, mainly due to the possibility that training was ineffective or the scenario that datasets are not representative. It is common to use validation methods to ensure that the model has been trained and tested in different scopes of data, observing how it behaves.

Cross-validation is a sampling method used to evaluate machine learning models on a limited dataset. The method takes a parameter that indicates the number of partitions. Each partition will be used as the validation set and the remaining folds to train a distinct model. The performance of all models is averaged. It is important to note that the partitions are stratified, as this reduces bias and variance compared to non-stratified models, as pointed out by (Kohavi, 1995).

3 RELATED WORK

Considering the number of resources spent searching for the ideal machine learning model, a deeper study is essential on how specific characteristics can help us reduce the number of possibilities.

In the work of (Thabtah et al., 2020), different sets of data with different degrees of imbalance were analyzed. For the analysis, Recall, Precision, and Accuracy asymmetries were observed for each scenario. Authors also tested balancing techniques such as SMOTE (Chawla et al., 2002), which generates new synthetic data using the K-Nearest Neighbors (KNN) algorithm. Five different datasets were used to conduct the study, named Cleveland, Credit-German, Diabetes, and Hepatitis. The largest is Credit-German,

with 1,000 instances and 20 features. The smallest is Hepatitis, with 155 instances and 19 features. The degree of imbalance varies for all datasets, reaching 9:1. During training, 10-fold cross-validation was performed, and the balance was changed to 1:1. The authors concluded that the fundamental behavior of the algorithm was to maximize the accuracy of the model. Furthermore, although the highest accuracy is concentrated in high imbalance experiments, these are biased and are not good classifiers.

In the work of (Oreski and Oreski, 2014), the performance of different classifiers on different datasets was analyzed. The SMOTE synthetic data generation technique was applied for each dataset. The classification algorithms employed were Neural Networks (Hagan et al., 1997), *Support Vector Machines* (Hearst et al., 1998), Repeated Incremental Pruning to Produce Error Reduction (RIPPER) (Park and Bae, 2015), and Naive Bayes (Zhang, 2005). Thirty different datasets were used, all obtained through the platform Knowledge Extraction based on Evolutionary Learning (KEEL) (Alcalá-Fdez et al., 2009). The proportion of instances in the majority and minority classes ranges from 9:1 to 41:1. The number of instances ranges from 92 to 1829. To evaluate the results, a paired t-test (Hsu and Lachenbruch, 2014) as well as a Wilcoxon signed-rank test (Bennett, 1964) were performed. Cross-validation was not used, and the hyperparameters of the classifiers were not fine-tuned. As a result, it was possible to observe a decrease in the performance of the algorithms after applying the SMOTE data balancing technique for the Naive Bayes algorithm. In contrast, a considerable increase was observed for the other classifiers when analyzing the AUC metric. However, when analyzing accuracy, the SMOTE technique did not positively contribute to the data sets' performance: the classifiers showed better performance when applied to the original, unbalanced data.

4 METHODOLOGY

In this section, the methodology adopted in this study is presented. We describe the datasets used and their particularities. The learning process that was performed for each algorithm and dataset and the the best hyperparameters for each model are detailed.

4.1 Datasets

To establish a solid basis for the present experiment, 15 datasets with different levels of unbalance, several instances, and several characteristics were selected.

Table 2: Description of the datasets used in the study.

ID	Name	#Inst	#Feat	#Prop
Eco	ecoli	336	7	8.6:1
Sat	satimage	6435	36	9.3:1
Aba	abalone	4177	10	9.7:1
Sic	sick_euthyroid	3163	42	9.8:1
Usc	us_crime	1994	100	12:1
Yea18	yeast_ml8	2417	103	13:1
Lib	libras_move	360	90	14:1
Arr	arrhythmia	452	278	17:1
Sol	solar_flare_m0	1389	32	19:1
Oil	oil	937	49	22:1
Car	car_eval_4	1728	21	26:1
Yea2	yeast_me2	1484	8	28:1
Wine	wine_quality	4898	11	26:1
Ozo	ozone_level	2536	72	34:1
Aba19	abalone_19	4177	10	130:1

Table 2 presents all data referring to the datasets used. The table columns indicate dataset name, the number of instances (#Inst) and features (#Feat), and also the proportion of the majority class to the minority class (Prop.) used for ordering the datasets.

All data were obtained from the open-source library imblearn (Lemaître et al., 2017). The selected datasets were structured to provide a basis for performance comparison, known as the term benchmark. Therefore, it is possible to see that after processing and formatting, a dataset generated more than one benchmark dataset, which is the case of the Abalone. In this study, the original dataset was used, as well as a variation called Abalone_19, where the difference between them was the formatting of the features to be classified.

4.2 Selection of Machine Learning Algorithms

Ten machine learning algorithms were selected. Five are base classifiers, namely: Naive Bayes (NB) (Zhang, 2005), Logistic Regression (LR) (Dong et al., 2016), Support Vector Machines (SVM) (Bhavan et al., 2019), K-Nearest Neighbors (Muliono et al., 2020) (KNN), and Decision Trees (DT) (Myles et al., 2004). The other five are ensembles: Stacking (Merz, 1999), Boosting (Schapire, 2013), Bagging (Ahmad et al., 2018), and Voting (Saqlain et al., 2019). Each algorithm has a drastically different operation.

The choice was made based on the assumption that different algorithms are more likely to generate models that perform different predictions, as they make decisions using different logic. Considering that each dataset is distinct, the use of several algorithms makes it more likely to generate at least

one base classifier with good performance. Using distinct base classification algorithms is also essential for building the Stacking because the diversity among classifiers generates models with greater generalizability and better performance (Kuncheva and Whitaker, 2003). The same logic was defined for the Voting-based algorithm.

4.3 Experimental Setup

Each ML algorithm has customizable hyperparameters, and they can impact the generated model in several ways, such as the learning rate, complexity, and weights for certain features. In our experiment evaluation, several classifiers were used, each with its specific hyperparameters. As there are many, the technique known as Grid Search (Liashchynskiy and Liashchynskiy, 2019) (GS) was used, which combines all of them, generating N models. The number of distinct hyperparameter combinations defines N. When searching for the hyperparameters, the function returns the model with the best performance considering an evaluation metric. Each generated model was evaluated using Cross-Validation.

The best models fitted by the base classification algorithms are compared using F1-Score since it relates two very relevant metrics in unbalanced scenarios: Precision and Recall. However, the other metrics presented in Section 2.3 are also generated for further analysis. Then, the same process is applied to the ensembles. Evaluation metrics are generated, and, finally, the performance of the best ensemble is analyzed. Results are generated for each dataset, and in the end, it is possible to compare the performance of the best base classifier and the best ensemble.

To create the Stacking algorithm, we used the estimators obtained using the Grid Search in each base algorithm used: LR, DT, SVM, and KNN. These are then combined in the meta-classifier, which the best is selected based on performance of the base classifiers.

Finally, we present the 15 unbalanced datasets selected for this study in Table 2. Where ID is related with the identification of the dataset, used in the obtained results, Name is the complete name of the dataset, #Inst the number of instances, #Feat the number of features and #Prop the proportion of examples.

5 RESULTS

In this section, we present the results and discuss them. To do so, we provide them in Table 3, which is divided into two parts. The first one presents the results related to the base classifiers and the second

to the ensemble. If an obtained value in the ensemble part is superior to the base classifier, we highlight it with **boldface**. Considering the structure of this table, rows are related to the different datasets. We have provided results considering the original datasets and their versions where the undersample (.Under) and oversample (.Over) were applied. Additionally, the columns represent the different evaluation metrics for the best-tuned model. The base model column shows the acronym of the best base classification algorithm. For F1, precision (Prec), and recall (Rec), we provide the results per class (C0 and C1) and its averaging (Avg). The last two columns are related to the Area Under the Curve (AUC) and Accuracy (Acc).

Our first analysis takes into account the accuracy. It is necessary to highlight that the base classifiers are already achieving high accuracy, therefore the usage of an ensemble can not provide a significant performance. Moreover, as mentioned before, consider the accuracy for an unbalanced problem does not seem a good alternative. For this reason, we are focusing on averaging F1-Score as the general evaluation metric.

Up to this point, we analyze in which cases ensemble outperformed the base classifier. Considering all 15 datasets in their standard versions, i.e., without applying over or undersampling, the ensembles performed best considering averaging F1-Score in four of them (26%): Oil, Sic, Spec, and Yea2. In the oversampling scenario, this rate remained the same (Aba19, Eco, Lib, and Yea2). Regarding undersample, ensembles were best in 8 datasets (53%): Aba, Aba19, Oil, Ozo, Sat, Sic, Spec, and Win.

Although almost all experiments presented better results using balancing techniques, the degree of imbalance of the standard datasets did not show a correlation with the improvement in F1-Score when applying these techniques. For instance, despite Aba.19 being the most imbalanced dataset (130:1), it reached a smaller F1 gain when compared with its original version, Abalone, which presents a distribution of 9.7:1.

It is important to note that the difference between ensembles and base classifiers was insufficient to justify their use. Only two cases presented significant differences after over or undersampling (Eco = 4.6% and Ozo = 4.3%). Therefore, it is necessary to analyze whether few gains are essential for the application field. Furthermore, even in the scenario in which the ensemble overreach the best base classifier applying undersampling, it is necessary to verify in-depth that there was no loss of information in removing instances at random. Possibly, techniques that analyze the distribution of data points can provide a basis of study with less damage to data quality in some cases.

In a closer look at the averaging Recall, we have

Table 3: Evaluation metrics performed in the 15 datasets considering over and undersampling.

Dataset	Base Model	F1 Avg	F1 C0	F1 C1	Rec Avg	Rec C0	Rec C1	Prec Avg	Prec C0	Prec C1	AUC	Acc
Aba	KNN	0.482	0.949	0.014	0.503	0.907	0.100	0.502	0.996	0.008	0.502	0.903
Aba_Over	KNN	0.872	0.855	0.888	0.899	1.000	0.799	0.874	0.748	1.000	0.874	0.874
Aba_Under	SVM	0.791	0.764	0.819	0.816	0.892	0.740	0.795	0.672	0.918	0.795	0.795
Aba_19	KNN	0.498	0.996	0.000	0.496	0.992	0.000	0.500	1.000	0.000	0.500	0.992
Aba_19_Over	KNN	0.976	0.976	0.977	0.977	1.000	0.954	0.976	0.952	1.000	0.976	0.976
Aba_19_Under	DT	0.714	0.644	0.785	0.812	0.933	0.690	0.742	0.542	0.942	0.742	0.736
Arr	DT	0.780	0.976	0.584	0.808	0.977	0.640	0.780	0.977	0.583	0.780	0.956
Arr_Over	SVM	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
Arr_Under	DT	0.833	0.840	0.826	0.854	0.900	0.808	0.842	0.817	0.867	0.842	0.840
Car	SVM	0.990	0.999	0.980	0.999	0.999	1.000	0.983	1.000	0.967	0.983	0.999
Car_Over	SVM	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
Car_Under	NB	0.992	0.992	0.992	0.993	1.000	0.986	0.993	0.986	1.000	0.993	0.992
Eco	KNN	0.802	0.964	0.639	0.856	0.958	0.753	0.793	0.970	0.617	0.793	0.934
Eco_Over	DT	0.943	0.940	0.947	0.949	0.996	0.902	0.943	0.890	0.997	0.943	0.943
Eco_Under	KNN	0.925	0.921	0.930	0.938	0.975	0.902	0.925	0.883	0.967	0.925	0.929
Lib	LR	0.877	0.990	0.763	0.940	0.980	0.900	0.850	1.000	0.700	0.850	0.981
Lib_Over	DT	0.994	0.994	0.994	0.994	1.000	0.989	0.994	0.988	1.000	0.994	0.994
Lib_Under	NB	0.832	0.807	0.857	0.850	0.783	0.917	0.867	0.867	0.867	0.867	0.850
Oil	DT	0.700	0.976	0.425	0.742	0.973	0.510	0.694	0.979	0.410	0.694	0.954
Oil_Over	SVM	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
Oil_Under	LR	0.836	0.834	0.839	0.847	0.827	0.867	0.840	0.855	0.825	0.840	0.842
Ozo	DT	0.540	0.981	0.099	0.557	0.973	0.141	0.536	0.989	0.082	0.536	0.963
Ozo_Over	SVM	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
Ozo_Under	LR	0.761	0.756	17.357	0.785	0.833	0.738	0.764	0.711	0.818	0.764	0.768
Sat	KNN	0.811	0.966	0.655	0.840	0.958	0.723	0.788	0.975	0.601	0.788	0.938
Sat_Over	SVM	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
Sat_Under	KNN	0.870	0.859	0.881	0.882	0.938	0.826	0.872	0.796	0.947	0.872	0.871
Sic	DT	0.921	0.985	0.857	0.925	0.985	0.864	0.920	0.986	0.854	0.920	0.973
Sic_Over	SVM	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
Sic_Under	DT	0.944	0.943	0.944	0.946	0.954	0.938	0.943	0.935	0.952	0.943	0.944
Sol	NB	0.619	0.964	0.275	0.626	0.963	0.289	0.621	0.964	0.279	0.621	0.931
Sol_Over	SVM	0.925	0.923	0.927	0.927	0.950	0.903	0.925	0.898	0.952	0.925	0.925
Sol_Under	SVM	0.754	0.736	0.773	0.777	0.826	0.727	0.760	0.679	0.840	0.760	0.758
Spec	KNN	0.878	0.983	0.774	0.972	0.968	0.975	0.824	0.998	0.650	0.824	0.968
Spec_Under	LR	0.910	0.917	0.902	0.921	0.887	0.955	0.913	0.960	0.865	0.913	0.911
Usc	LR	0.743	0.970	0.516	0.834	0.954	0.715	0.700	0.986	0.413	0.700	0.943
Usc_Over	SVM	0.994	0.994	0.994	0.994	1.000	0.989	0.994	0.989	1.000	0.994	0.994
Usc_Under	NB	0.873	0.874	0.871	0.880	0.864	0.897	0.873	0.893	0.853	0.873	0.873
Win	DT	0.570	0.981	0.159	0.716	0.966	0.467	0.547	0.996	0.099	0.547	0.962
Win_Over	SVM	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
Win_Under	LR	0.709	0.717	0.701	0.714	0.702	0.726	0.710	0.738	0.683	0.710	0.710
Yea2	DT	0.649	0.981	0.318	0.694	0.975	0.413	0.630	0.987	0.273	0.630	0.962
Yea2_Over	KNN	0.946	0.943	0.949	0.952	1.000	0.904	0.947	0.893	1.000	0.947	0.947
Yea2_Under	SVM	0.831	0.832	0.830	0.839	0.833	0.845	0.833	0.840	0.827	0.833	0.832
Yea18	NB	0.496	0.926	0.065	0.495	0.926	0.065	0.498	0.927	0.068	0.498	0.864
Yea18_Over	SVM	0.997	0.997	0.997	0.997	1.000	0.995	0.997	0.995	1.000	0.997	0.997
Yea18_Under	KNN	0.578	0.573	0.584	0.580	0.587	0.573	0.579	0.562	0.597	0.579	0.579
Mean	-	0.833	0.920	1.100	0.853	0.939	0.769	0.831	0.911	0.751	0.831	0.917

Dataset	Ensemble Model	F1 Avg	F1 C0	F1 C1	Rec Avg	Rec C0	Rec C1	Prec Avg	Prec C0	Prec C1	AUC	Acc
Aba	AdaBoost	0.478	0.951	0.005	0.503	0.907	0.100	0.501	1.000	0.003	0.501	0.907
Aba_Over	AdaBoost	0.859	0.843	0.875	0.880	0.964	0.795	0.861	0.750	0.972	0.861	0.861
Aba_Under	Random Forest	0.796	0.773	0.819	0.815	0.880	0.749	0.799	0.693	0.906	0.799	0.799
Aba_19	Random Forest	0.498	0.996	0.000	0.496	0.992	0.000	0.500	1.000	0.000	0.500	0.992
Aba_19_Over	AdaBoost	0.998	0.998	0.998	0.998	1.000	0.995	0.998	0.995	1.000	0.998	0.998
Aba_19_Under	Extra Trees	0.735	0.682	0.787	0.828	0.927	0.730	0.763	0.608	0.917	0.763	0.752
Arr	AdaBoost	0.753	0.977	0.530	0.766	0.975	0.557	0.764	0.979	0.550	0.764	0.956
Arr_Over	Random Forest	0.998	0.998	0.998	0.998	1.000	0.995	0.998	0.995	1.000	0.998	0.998
Arr_Under	AdaBoost	0.791	0.790	0.792	0.813	0.850	0.775	0.800	0.767	0.833	0.800	0.800
Car	AdaBoost	0.954	0.997	0.911	0.997	0.994	1.000	0.921	1.000	0.843	0.921	0.994
Car_Over	AdaBoost	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
Car_Under	Random Forest	0.992	0.992	0.992	0.993	1.000	0.986	0.993	0.986	1.000	0.993	0.992
Eco	AdaBoost	0.769	0.957	0.581	0.826	0.952	0.700	0.765	0.963	0.567	0.765	0.923
Eco_Over	AdaBoost	0.983	0.983	0.984	0.984	1.000	0.969	0.983	0.967	1.000	0.983	0.983
Eco_Under	AdaBoost	0.899	0.899	0.899	0.904	0.892	0.917	0.904	0.917	0.892	0.904	0.900
Lib	Random Forest	0.872	0.987	0.757	0.987	0.974	1.000	0.817	1.000	0.633	0.817	0.975
Lib_Over	Random Forest	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
Lib_Under	Extra Trees	0.832	0.807	0.857	0.850	0.783	0.917	0.867	0.867	0.867	0.867	0.850
Oil	AdaBoost	0.721	0.980	0.461	0.786	0.973	0.600	0.689	0.988	0.390	0.689	0.962
Oil_Over	AdaBoost	0.998	0.998	0.998	0.998	1.000	0.996	0.998	0.996	1.000	0.998	0.998
Oil_Under	AdaBoost	0.850	0.852	0.847	0.868	0.853	0.883	0.853	0.875	0.830	0.853	0.853
Ozo	Random Forest	0.493	0.985	0.000	0.486	0.971	0.000	0.500	1.000	0.000	0.500	0.971
Ozo_Over	AdaBoost	0.997	0.997	0.997	0.997	1.000	0.994	0.997	0.993	1.000	0.997	0.997
Ozo_Under	Random Forest	0.796	0.794	0.798	0.799	0.809	0.790	0.797	0.784	0.811	0.797	0.797
Sat	AdaBoost	0.789	0.964	0.613	0.838	0.951	0.725	0.757	0.978	0.536	0.757	0.935
Sat_Over	AdaBoost	0.942	0.939	0.945	0.947	0.992	0.902	0.942	0.892	0.993	0.942	0.942
Sat_Under	Random Forest	0.872	0.865	0.880	0.879	0.915	0.842	0.873	0.823	0.923	0.873	0.873
Sic	AdaBoost	0.941	0.989	0.892	0.945	0.988	0.902	0.937	0.990	0.884	0.937	0.980
Sic_Over	AdaBoost	0.990	0.990	0.991	0.991	0.999	0.982	0.990	0.982	0.999	0.990	0.990
Sic_Under	AdaBoost	0.945	0.946	0.944	0.947	0.945	0.949	0.945	0.949	0.941	0.945	0.945
Sol	AdaBoost	0.556	0.974	0.138	0.690	0.955	0.425	0.540	0.994	0.086	0.540	0.950
Sol_Over	AdaBoost	0.848	0.840	0.855	0.853	0.890	0.816	0.848	0.797	0.899	0.848	0.848
Sol_Under	Extra Trees	0.735	0.745	0.725	0.766	0.763	0.769	0.744	0.767	0.721	0.744	0.743
Spec	Extra Trees	0.890	0.985	0.796	0.969	0.972	0.967	0.849	0.998	0.700	0.849	0.972
Spec_Under	Random Forest	0.931	0.933	0.928	0.944	0.947	0.942	0.928	0.930	0.925	0.928	0.933
Usc	AdaBoost	0.711	0.967	0.454	0.802	0.949	0.654	0.670	0.986	0.353	0.670	0.938
Usc_Over	AdaBoost	0.989	0.989	0.989	0.989	1.000	0.978	0.989	0.978	1.000	0.989	0.989
Usc_Under	Extra Trees	0.843	0.837	0.848	0.849	0.875	0.823	0.843	0.807	0.880	0.843	0.843
Win	AdaBoost	0.536	0.982	0.090	0.807	0.964	0.650	0.525	1.000	0.050	0.525	0.964
Win_Over	AdaBoost	0.968	0.967	0.969	0.969	0.991	0.947	0.968	0.945	0.991	0.968	0.968
Win_Under	Random Forest	0.775	0.786	0.763	0.783	0.754	0.811	0.776	0.825	0.727	0.776	0.776
Yea2	AdaBoost	0.668	0.982	0.353	0.787	0.975	0.600	0.632	0.990	0.273	0.632	0.966
Yea2_Over	AdaBoost	0.989	0.989	0.989	0.989							

a mean of 0.85 for all the ensembles and base classifiers. Comparing the achieved results, we have that for 21 different datasets, out of the 47 considered, we have that the achieved results is superior or equal than the ones obtained in the base classifiers. We also have that both balancing techniques improved the obtained values for the dataset Aba_19, Car and Lib. On the other hand, neither approach holds this behavior for the datasets Aba, Arr, Sat, Sol, Usc, and Yea18. For the remaining cases, at least one technique, over or undersampling, enhances the results.

A similar analysis was performed with the averaging Precision, for both the base classifier and the ensemble. Once again, we can notice a similar behavior in these cases, which means around 0.83. We have 20 different cases presenting best Precision scores than the base classifiers. Considering the datasets, we have that the sampling approaches did not increase the results only for Arr, Sol, and Usc.

The mean AUC regarding all datasets for the base classifiers was 0.831, while the achieved mean for the ensembles was 0.823. Additionally, for 20 different datasets, the ensembles AUC outperformed or tied the base classifiers. Again, for Arr, Sol, and Usc, this metric was worst than base classifiers. In all other cases, the usage of balancing demonstrated at least one situation better than those in base classifiers.

Finally, analyzing the performance of the ensembles and base classifiers, it is possible to see that, even for cases where there is a significant imbalance, there is no direct relationship that allows us to perform a prediction based on this variable.

6 CONCLUSION

In this paper, we study the performance relationship of ensembles and base classifiers from the perspective of unbalance. We collect several metrics such as accuracy, recall, precision, and F1-score. We analyzed whether we can fit ensembles with outstanding predictive capacity than base classifiers in scenarios of greater imbalance.

After analyzing the experimental results over fifteen datasets, it is possible to state that there is no direct and strong correlation between imbalance and the performance of ensembles. In most cases, we had base classifiers that performed better. These results added to the extra resources spent for selection and the classifiers' training time, helping us to conclude that it is not reasonable to use ensembles just based on an imbalance distribution. The problem presents itself to the authors as a mixture of possible and previously analyzed variables, in addition to other measures that

were possibly not analyzed.

By applying balancing techniques, we got slightly different results. For oversampling, we got the same rate of scenarios where ensembles outperformed the base classifiers, and for undersampling, that rate was 53%. It was impossible to observe a correlation between the degree of imbalance in the dataset and how this distribution can benefit the ensemble. In addition to the results, it is also necessary to analyze how the sampling method selects data to be cloned or excluded, as it can drastically affect the representativeness of the studied phenomenon.

ACKNOWLEDGEMENTS

This study was supported by CNPq (305805/2021-5) and PNPd/CAPES (464880/2019-00).

REFERENCES

- Ahmad, M. W., Reynolds, J., and Rezgui, Y. (2018). Predictive modelling for solar thermal energy systems: A comparison of support vector regression, random forest, extra trees and regression trees. *Journal of cleaner production*, 203:810–821.
- Alcalá-Fdez, J., Sánchez, L., García, S., Del Jesus, M., Ventura, S., Garrell, J., Otero, J., Romero, C., Bacardit, J., Rivas, V., et al. (2009). Knowledge extraction based on evolutionary learning. *Reference manual*, 0–31.
- Barboza, F., Kimura, H., and Altman, E. (2017). Machine learning models and bankruptcy prediction. *Expert Systems with Applications*, 83:405–417.
- Bennett, B. (1964). A bivariate signed rank test. *Journal of the Royal Statistical Society: Series B (Methodological)*, 26(3):457–461.
- Bhavan, A., Chauhan, P., Shah, R. R., et al. (2019). Bagged support vector machines for emotion recognition from speech. *Knowledge-Based Systems*, 184:104886.
- Bishop, C. M. (2006). Pattern recognition. *Machine learning*, 128(9).
- Chawla, N. V., Bowyer, K. W., Hall, L. O., and Kegelmeyer, W. P. (2002). Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357.
- Dietterich, T. G. et al. (2002). Ensemble learning. *The handbook of brain theory and neural networks*, 2(1):110–125.
- Dong, L., Wesseloo, J., Potvin, Y., and Li, X. (2016). Discrimination of mine seismic events and blasts using the fisher classifier, naive bayesian classifier and logistic regression. *Rock Mechanics and Rock Engineering*, 49(1):183–211.
- Garcia, N., Tiggeman, F., Borges, E., Lucca, G., Santos, H., and Dimuro, G. (2021). Exploring the relationships between data complexity and classification diversity

- in ensembles. In *Proceedings of the 23rd International Conference on Enterprise Information Systems*, pages 652–659. INSTICC, SciTePress.
- Hagan, M. T., Demuth, H. B., and Beale, M. (1997). *Neural network design*. PWS Publishing Co.
- Hart, P. E., Stork, D. G., and Duda, R. O. (2000). *Pattern classification*. Wiley Hoboken.
- Hawkins, D. M. (2004). The problem of overfitting. *Journal of chemical information and computer sciences*, 44(1):1–12.
- Haykin, S. and Network, N. (2004). A comprehensive foundation. *Neural networks*, 2(2004):41.
- Hearst, M. A., Dumais, S. T., Osuna, E., Platt, J., and Scholkopf, B. (1998). Support vector machines. *IEEE Intelligent Systems and their applications*, 13(4):18–28.
- Hoo, Z. H., Candlish, J., and Teare, D. (2017). What is an roc curve?
- Hsu, H. and Lachenbruch, P. A. (2014). Paired t test. *Wiley StatsRef: statistics reference online*.
- Ishibuchi, H., Nakashima, T., and Nii, M. (2004). *Classification and modeling with linguistic information granules: Advanced approaches to linguistic Data Mining*. Springer Science & Business Media.
- Japkowicz, N. (2000). The class imbalance problem: Significance and strategies. In *Proc. of the Int'l Conf. on Artificial Intelligence*, volume 56. Citeseer.
- Japkowicz, N. and Stephen, S. (2002). The class imbalance problem: A systematic study. *Intelligent data analysis*, 6(5):429–449.
- Kohavi, R. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 2, IJCAI'95*, page 1137–1143, San Francisco, CA, USA. Morgan Kaufmann PUBLISHERS Inc.
- Kourou, K., Exarchos, T. P., Exarchos, K. P., Karamouzis, M. V., and Fotiadis, D. I. (2015). Machine learning applications in cancer prognosis and prediction. *Computational and structural biotechnology journal*, 13:8–17.
- Kuncheva, L. I. and Whitaker, C. J. (2003). Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. *Machine learning*, 51(2):181–207.
- Lemaître, G., Nogueira, F., and Aridas, C. K. (2017). Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning. *The Journal of Machine Learning Research*, 18(1):559–563.
- Liashchynskiy, P. and Liashchynskiy, P. (2019). Grid search, random search, genetic algorithm: a big comparison for nas. *arXiv preprint arXiv:1912.06059*.
- Merz, C. J. (1999). Using correspondence analysis to combine classifiers. *Machine Learning*, 36(1-2):33–58.
- Mosavi, A., Ozturk, P., and Chau, K.-w. (2018). Flood prediction using machine learning models: Literature review. *Water*, 10(11):1536.
- Muliono, R., Lubis, J. H., and Khairina, N. (2020). Analysis k-nearest neighbor algorithm for improving prediction student graduation time. *Sinkron: jurnal dan penelitian teknik informatika*, 4(2):42–46.
- Myles, A. J., Feudale, R. N., Liu, Y., Woody, N. A., and Brown, S. D. (2004). An introduction to decision tree modeling. *Journal of Chemometrics: A Journal of the Chemometrics Society*, 18(6):275–285.
- Opitz, D. and Maclin, R. (1999a). Popular ensemble methods: An empirical study. *Journal of artificial intelligence research*, 11:169–198.
- Opitz, D. and Maclin, R. (1999b). Popular ensemble methods: An empirical study. *Journal of Artificial Intelligence Research*, 11:169–198.
- Oreski, G. and Oreski, S. (2014). An experimental comparison of classification algorithm performances for highly imbalanced datasets. In *Central European Conference on Information and Intelligent Systems*, page 4. Faculty of Organization and Informatics Varazdin.
- Park, B. and Bae, J. K. (2015). Using machine learning algorithms for housing price prediction: The case of fairfax county, virginia housing data. *Expert systems with applications*, 42(6):2928–2934.
- Polat, K., Yosunkaya, Ş., and Güneş, S. (2008). Comparison of different classifier algorithms on the automated detection of obstructive sleep apnea syndrome. *Journal of Medical Systems*, 32(3):243–250.
- Quinlan, J. R. (1986). Induction of decision trees. *Machine learning*, 1(1):81–106.
- Quinlan, J. R. et al. (1996). Bagging, boosting, and c4. 5. In *Aaai/iaai, Vol. 1*, pages 725–730.
- Saqlain, M., Jargalsaikhan, B., and Lee, J. Y. (2019). A voting ensemble classifier for wafer map defect patterns identification in semiconductor manufacturing. *IEEE Transactions on Semiconductor Manufacturing*, 32(2):171–182.
- Schapire, R. E. (2013). Explaining adaboost. In *Empirical inference*, pages 37–52. Springer.
- Steinwart, I. and Christmann, A. (2008). *Support vector machines*. Springer Science & Business Media.
- Thabtah, F., Hammoud, S., Kamalov, F., and Gonsalves, A. (2020). Data imbalance in classification: Experimental evaluation. *Information Sciences*, 513:429–441.
- Visa, S., Ramsay, B., Ralescu, A. L., and Van Der Knaap, E. (2011). Confusion matrix-based feature selection. *MAICS*, 710:120–127.
- Yap, B. W., Abd Rani, K., Abd Rahman, H. A., Fong, S., Khairudin, Z., and Abdullah, N. N. (2014). An application of oversampling, undersampling, bagging and boosting in handling imbalanced datasets. In *Proceedings of the first international conference on advanced data and information engineering (DaEng-2013)*, pages 13–22. Springer.
- Zhang, H. (2005). Exploring conditions for the optimality of naive bayes. *International Journal of Pattern Recognition and Artificial Intelligence*, 19(02):183–198.