Cost-Sensitive Algorithms for Imbalanced Text Classification: a study case in Brazilian legal domain

Anonymous ACL submission

Abstract

This article discusses the challenges of imbalanced classification in machine learning, where algorithms often wrongly assume an even distribution of instances across classes. This issue is common in real-world scenarios, leading to poor representation of minority classes in training data. To combat this, Cost-Sensitive Learning techniques have been developed, focusing on minimizing the overall misclassification cost rather than merely optimizing accuracy. These techniques are categorized into three types: Cost-Sensitive Resampling, Algorithms, and Hybrid techniques. The research presents a case study on classifying lawsuits into repetitive themes in São Paulo Court, Brazil, using these cost-sensitive approaches on an imbalanced dataset. The goal is to automate the classification of lawsuits to save time, use human resources more effectively, and speed up lawsuit resolution. The study highlights the effectiveness of cost-sensitive techniques in handling imbalanced classification and their benefits in real-world applications, particularly in the legal field, by enhancing efficiency and reducing manual workload and processing time for lawsuits.

1 Introduction

The vast majority of machine learning (ML) algorithms designed for classification tasks assume an equal distribution of instances across the observed classes. However, this assumption only holds in some practical scenarios. In most real world situations, classification datasets are imbalanced. Imbalanced classification, characterized by skewed class distributions, poses significant challenges in applied ML, primarily due to the assumption of balanced class distribution and uniform prediction errors made by classifiers, encompassing false negatives and false positives. Consequently, accurately identifying instances from the minority class becomes crucial, along with addressing the under-representation of the minority class in training data, making Imbalanced classification stand out as one of the most challenging predicaments in ML (Thai-Nghe et al., 2010).

We can deal with imbalanced classification tasks using several conceptualizations and techniques developed and employed under the umbrella of Cost-Sensitive Learning (CSL), which considers the costs associated with prediction errors, among other costs, during the training of a ML model. These costs represent the penalties incurred from incorrect predictions. In CSL, instead of classifying instances as correct or incorrect, a misclassification cost is assigned to each class or instance. Therefore, the objective shifts from optimizing accuracy to minimizing the total misclassification cost (Ma and He, 2013). The primary aim of CSL is to reduce the overall cost incurred by a model during its training, assuming that different types of prediction errors have distinct and known associated costs (Sammut and Webb, 2011).

In many previous studies, researchers have focused on modifying the internal structure of conventional classification procedures to adjust the algorithm's sensitivity towards the larger class. These efforts aimed to address the challenges posed by class imbalance. On the other hand, some authors have taken a different approach by proposing novel methods to alleviate the imbalanced class distribution. Their work has explored alternative strategies to tackle the class imbalance problem (Zhao et al., 2011; Galar et al., 2011; Pang et al., 2013; Dai, 2015).

In this work, we present a case study that applies and compares Cost-Sensitive techniques on an imbalanced dataset of legal texts from the São Paulo Court (Tribunal de Justiça de São Paulo or TJSP) in Brazil. The case study focuses on the classification of lawsuits into repetitive themes, which refer to sets of lawsuits on appeal sharing identical legal arguments based on similar questions of law. At TJSP, civil servants face the daily challenge of manually reading numerous lengthy lawsuit decisions to determine their classification as repetitive. By automating the classification of repetitive themes, the process can be expedited, saving time and facilitating faster resolution of lawsuits (de Justiça Departamento de Pesquisas Judiciárias, 2022 [Online].).

The rest of this paper is organised as follows: Section 2 presents a background and a brief description of techniques and methodologies employed; Section 3 describes the environment of repetitive theme in legal domain and formulates the problem; Section 4 reports experiments; and, Section 5 presents our conclusions.

2 Background

This section discusses three Cost-Sensitive Learning (CSL) approaches for imbalanced learning: Cost-Sensitive Resampling, Cost-Sensitive Algorithms, and Cost-Sensitive Hybrid approaches, as detailed in (Elkan, 2001). These methods modify training set composition or algorithm parameters to address the imbalances in classification tasks.

2.1 Cost-Sensitive Algorithms

Cost-Sensitive Algorithms involve incorporating cost matrices into machine learning (ML) algorithms. This requires unique modifications for each algorithm, such as the Support Vector Machine (SVM), Decision Tree (DT), and Logistic Regression (LR) (Quinlan, 1986; Hearst et al., 1998; M.D., 1944).

2.1.1 Cost-Sensitive SVM

The SVM algorithm can be adapted for imbalanced datasets by modifying the margin, making it cost-sensitive. This version, termed weighted SVM, uses a regularization hyperparameter to prioritize the minority class (Yang et al., 2007; Kuhn et al., 2013).

2.1.2 Cost-Sensitive DT

For Decision Trees, modifying the criteria for evaluating split points can help address class imbalances, creating a weighted decision tree.

2.1.3 Cost-Sensitive LR

Logistic Regression can be adapted by adjusting the weights during the training phase, resulting in a cost-sensitive version better suited for imbalanced classification tasks.

2.1.4 Cost-Sensitive XGBoost

XGBoost can be made cost-sensitive by adjusting the ‘scale_pos_weight‘ hyperparameter, improving its performance on imbalanced datasets.

2.1.5 Cost-Sensitive RF

Random Forest (RF) can be adapted to imbalanced problems by weighting classes during the calculation of the impurity score for split points, resulting in a Weighted Random Forest (Chen et al., 2004).

2.2 Random Sampling

Random sampling techniques, including oversampling and undersampling, adjust the class distribution in the training set. These methods are simple and can be effective, but their efficiency varies based on the dataset and model (Ma and He, 2013).

2.2.1 Random Oversampling

Random Oversampling can influence models like SVMs and decision trees. However, it risks overfitting in severely imbalanced datasets (Branco et al., 2015).

2.2.2 Random Undersampling

Random Undersampling is suitable when sufficient minority class examples exist. It involves reducing instances from the majority class, but may discard valuable data.

3 Problem Formulation

In this section, we describe the context of our study case, a real-world classification problem found in TJSP (Tribunal de Justiça de São Paulo), and after, we formulate the mathematical problem and our fitness function.

3.1 Context

In Brazilian justice courts, a mechanism of Repetitive Appeal enables the simultaneous adjudication of multiple special appeals that address an identical legal dispute. Through sampling, specific cases are chosen and forwarded to the Superior Court of Justice, in Brazil, for allocation. Subsequently, all cases about the same subject matter are put on hold until the resolution of the repetitive appeal. Once the repetitive appeal is decided, the ruling is consolidated as a “Repetitive Theme”. Upon publication of the decision regarding the repetitive issue, it becomes applicable to the other suspended proceedings. It is important to note that the numbering of these cases may not follow a sequential order, as the theme is either pending judgment or invalidated.

Our study case comprises a lawsuit classification into repetitive theme 929. Theme 929 discusses the hypotheses for applying the double repetition provided in art. 42, sole paragraph, of Federal Law n. 8.078/1990, also known as the Consumer Defense Code, namely:

> *Art. 42. In debt collection, the defaulting consumer will not be exposed to ridicule, nor will he be subjected to any embarrassment or threat.*
>
> *Single paragraph. The consumer charged an undue amount has the right to repeat the undue amount for an amount equal to twice what he paid in excess, plus monetary correction and legal interest, except in the case of a justifiable mistake.*

The training set has 13,570 texts of lawsuit decisions, through which one should infer whether the lawsuit belongs to repetitive theme 929 (class_1) or not (class_0). This is an imbalanced classification problem, given that 2,088 lawsuits belong to the theme repetitive 929 and 11,482 do not. So, the class distribution of the training set is about a 1:5 ratio for the minority class to the majority class. The validation set has 2,560 texts of lawsuit decisions, 584 of them of theme 929 (class_1) and 1,976 not (class_0). In this case, the class distribution of the validation set is about a 1:3 ratio for the minority class to the majority class. Figure 1 shows class distributions of the training and validation sets.

On the one hand, we want to correct the rare repetitive themes (true positives) of the production dataset; on the other hand, we do not want to erroneously point out legal cases as repetitive themes (false positives) since the judicial process has its procedure suspended.

3.2 Mathematical Formulation

In CSL, a “cost” is a penalty associated with an incorrect prediction. Minimizing the total cost is the primary goal when using CSL to train a predictive model, by assuming that different types of prediction errors have different and known associated costs. Applying CSL for imbalanced classification problems concerns assigning different costs associated with the different misclassification errors to then using specialized methods to take those costs into account. A cost matrix helps to understand the misclassification of miscellaneous costs.

A confusion matrix summarises the classifications made by a model for each class, separated by the class to which each instance belongs. Cell valour in the table refers to the number of samples corresponding to respective rows and columns. The hits are True Positives (TP) and True Negatives (TN). The errors are False Positives (FP) and False Negatives (FN). By assigning a cost to each of the cells of the confusion matrix, we have a cost matrix (Elkan, 2001). Table 1 shows a confusion matrix for a binary classification, where the columns refer to the actual classes, and the rows refer to the predicted classes. We use the notation $C(i, j)$ to indicate the cost, the first value, i, represented as the predicted class and the second value, j, represents the actual class.

Table 1: Cost Matrix

	Actual Negative	**Actual Positive**
Predicted Negative	C(0,0), TN	C(0,1), FN
Predicted Positive	C(1,0), FP	C(1,1), TP

Equation 1 calculates the total cost of misclassification errors. Depending on the problem domain, the cost can be given by a simple function or a complex multi-dimensional function, including monetary costs, reputation costs, and more. For imbalanced classification tasks, the most straightforward approach is to consider $C(i, j)$ as constant, assigning costs based on the inverse class distribution. For our training set, with a 1 to 5 ratio of instances in the minority class to the majority class, the cost of misclassification errors can be the

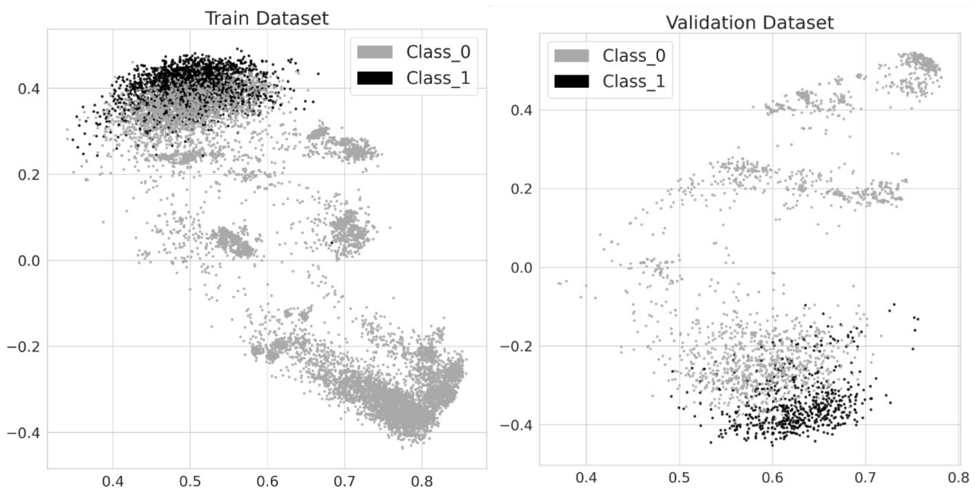

Figure 1: Lawsuit Decisions Sets With Class Imbalance.

inverse: the cost of a False Negative is five and the cost of a False Positive is 1.

$$TotalCost = C(0,1) * FN + C(1,0) * FP \quad (1)$$

Thus, the problem can be formulated as: *to find out the cost-sensitive technique that better minimises the total cost of misclassification errors in the repetitive theme 929.*

4 Study Case

In this section, we report our experiment with a real study case from lawsuit decisions classification from Court São Paulo. These decisions are classified into two distinct categories: classifying whether a lawsuit belongs to a collection of lawsuits that pertain to the repetitive theme (class_1), or if it does not (class_0).Our primary objective is to address the research query: *which learning cost-sensitive algorithm better minimize the total misclassification cost in lawsuit decisions classification task using our imbalanced dataset?*

Our imbalanced dataset, described in Section3, was vectorization by TF-IDF technique(Salton and Buckley, 1988), using 3,000 features, formatted by n-grams with n ranging from 1 to 5. We use a transformer to perform linear dimensionality reduction by means of truncated singular value decomposition, in order to show how the lawsuits decisions texts of both classes are spreed in our dataset. We compare five classifiers: Support Vector Machine (SVM), Decision Tree (DT), Logistic Regression (LR), Extreme Gradient Boosting algorithm (XGBoost) and Random Forest (RF).

4.1 Weightings Tuning

We use the scikit-learn Python ML library (Pedregosa et al., 2011) to implement the cost-sensitive SVM, DT, and LR. This library provide the class_weight argument that defines each class label and the weighting to apply to the λ value in the calculation of the soft margin. In case of XGBoost (Chen and Guestrin, 2016), scikit-learn fits the scale_pos_weight hyperparameter to to implement the cost-sensitive XGBoost. We use repeated cross-validation to evaluate the model, with three repeats of 10-fold cross-validation. The mode performance is reported using the mean ROC area under curve (ROC AUC) averaged over repeats and all folds. We test a range of different class weightings for weighted SVM, DT, and LR to find out which results in the best ROC AUC score.

Table 2 shows the results in the ROC AUC score to each class_weightg and scale_pos_weight for weighted SVM, DT, LR, XGBoost, and RF. For SVM, DT, and LR algorithms, the best ROC AUC score was with class weightings {0: 5, 1: 1}, whereas for RF, with class weightings {0: 1, 1: 1}. The weighted XGBoost achieved its best ROC AUC score using scale_pos_weight equal 5. Considering all ROC AUC scores, the best was 0.999107 with a standard deviation of 0.000807 using SVM, followed by 0.999083 with a standard deviation of 0.000984 using LR, 0.998924 with a standard deviation of 0.001016 using XGBoost, 0.998797 with standard deviation 0.000956 using RF, and, lastly, DT achieved 0.968991 in ROC AUC score with standard deviation 0.005106.

Table 2: ROC AUC Score of Cost-Sensitive Algorithms.

Algor.	Mean	SD	Weightings
	0.999107	0.000807	class_weight:{0: 5.0, 1: 1.0}
	0.999101	0.000809	class_weight:{0: 2.5, 1: 1.0}
SVM	0.999061	0.000836	class_weight:{0: 1.0, 1: 1.0}
	0.999064	0.000827	class_weight:{0: 1.0, 1: 2.5}
	0.999064	0.000827	class_weight:{0: 1.0, 1: 5.0}
	0.968991	0.005106	class_weight:{0: 5.0, 1: 1.0}
	0.968107	0.007710	class_weight:{0: 2.5, 1: 1.0}
DT	0.967528	0.007746	class_weight:{0: 1.0, 1: 1.0}
	0.967547	0.007170	class_weight:{0: 1.0, 1: 2.5}
	0.963844	0.010183	class_weight:{0: 1.0, 1: 5.0}
	0.999083	0.000984	class_weight:{0: 5.0, 1: 1.0}
	0.999068	0.000961	class_weight:{0: 2.5, 1: 1.0}
LR	0.999027	0.000940	class_weight:{0: 1.0, 1: 1.0}
	0.999065	0.000946	class_weight:{0: 1.0, 1: 2.5}
	0.999079	0.000944	class_weight:{0: 1.0, 1: 5.0}
	0.998914	0.001245	scale_pos_weight:{1}
	0.998901	0.001185	scale_pos_weight:{2}
XGBoost	0.998898	0.001103	scale_pos_weight:{3}
	0.998919	0.001070	scale_pos_weight:{4}
	0.998924	0.001016	scale_pos_weight:{5}
	0.998695	0.000949	class_weight:{0: 5.0, 1: 1.0}
	0.998744	0.000915	class_weight:{0: 2.5, 1: 1.0}
RF	0.998797	0.000956	class_weight:{0: 1.0, 1: 1.0}
	0.998774	0.000957	class_weight:{0: 1.0, 1: 2.5}
	0.998771	0.000947	class_weight:{0: 1.0, 1: 5.0}

4.2 Model Training

Various performance metrics have been used for imbalance classification tasks. In this study, we use the most popular of them, balanced accuracy f1-score, geometric mean (g-mean) and Cohen's kappa (kappa) to assess the predictive performance of the imbalanced classification approaches using random resampling methods. We report the performance obtained for the training and validation sets.

First, we use train the models for the classification tasks using cost-sensitive algorithms, unsetting the class_weight and the scale_pos_weight hyperparameters, which provide the best ROC AUC score for each algorithm. After, we train the models for the classification tasks using random resampling methods. Afterwards, we evaluate performance metrics to classification using training and validation sets using cost-sensitive algorithms, c.f. shown in Table 3 and Table 4. Table 5 and Table 6 show metrics using the oversampling method. Table 7 and Table 8 show metrics using undersampling method. Table 9 and Table 10 show metrics using the combination of over and undersampling.

In the training set, all cost-sensitive algorithms tested achieved good predictive performance (above 99%), c.f. shown in Table 3. In the

Table 3: Cost-sensitive Algorithms - Training.

Algorithm	bal_acc	precision	recall	f1-score	g-mean	kappa
SVM	0.9990	0.9997	0.9997	0.9997	0.9990	0.9989
DT	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
LR	0.9930	0.9945	0.9945	0.9945	0.9930	0.9789
XGBoost	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
RF	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000

Table 4: Cost-sensitive Algorithms - Validation.

Algorithm	bal_acc	precision	recall	f1-score	g-mean	kappa
SVM	0.9811	0.9867	0.9867	0.9867	0.9811	0.9622
DT	0.9759	0.9783	0.9777	0.9779	0.9759	0.9377
LR	0.9842	0.9868	0.9867	0.9868	0.9841	0.9625
XGBoost	0.9876	0.9885	0.9883	0.9883	0.9876	0.9670
RF	0.9872	0.9870	0.9867	0.9868	0.9872	0.9627

Table 5: Oversampling Method - Training

Algorithm	bal_acc	precision	recall	f1-score	g-mean	kappa
SVM	0.9981	0.9982	0.9982	0.9982	0.9981	0.9932
DT	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
LR	0.9916	0.9933	0.9932	0.9932	0.9916	0.9742
XGBoost	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
RF	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000

Table 6: Oversampling Method - Validation.

Algorithm	bal_acc	precision	recall	f1-score	g-mean	kappa
SVM	0.9825	0.9878	0.9878	0.9878	0.9824	0.9655
DT	0.9764	0.9790	0.9785	0.9786	0.9764	0.9398
LR	0.9831	0.9853	0.9851	0.9852	0.9831	0.9582
XGB	0.9855	0.9879	0.9878	0.9879	0.9855	0.9658
RF	0.9871	0.9870	0.9867	0.9867	0.9871	0.9627

Table 7: Undersampling Method - Training.

Algorithm	bal_acc	precision	recall	f1-score	g-mean	kappa
SVM	0.9981	0.9982	0.9982	0.9982	0.9981	0.9932
DT	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
LR	0.9916	0.9933	0.9932	0.9932	0.9916	0.9742
XGBoost	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
RF	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000

Table 8: Undersampling Method - Validation.

Algorithm	bal_acc	precision	recall	f1-score	g-mean	kappa
SVM	0.9825	0.9878	0.9878	0.9878	0.9824	0.9655
DT	0.9764	0.9790	0.9785	0.9785	0.9764	0.9398
LR	0.9831	0.9853	0.9851	0.9851	0.9831	0.9582
XGB	0.9855	0.9879	0.9878	0.9878	0.9855	0.9658
RF	0.9871	0.9870	0.9867	0.9867	0.9871	0.9627

Table 9: Over and Undersampling Method - Training.

Algorithm	bal_acc	precision	recall	f1-score	g-mean	kappa
SVM	0.9981	0.9982	0.9982	0.9982	0.9981	0.9932
DT	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
LR	0.9916	0.9933	0.9932	0.9932	0.9916	0.9742
XGBoost	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
RF	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000

Table 10: Over and Undersampling Method - Validation.

Algorithm	bal_acc	precision	recall	f1-score	g-mean	kappa
SVM	0.9825	0.9878	0.9878	0.9878	0.9824	0.9655
DT	0.9764	0.9790	0.9785	0.9786	0.9764	0.9398
LR	0.9831	0.9853	0.9851	0.9852	0.9831	0.9582
XGB	0.9855	0.9879	0.9878	0.9879	0.9855	0.9658
RF	0.9871	0.9870	0.9867	0.9867	0.9871	0.9627

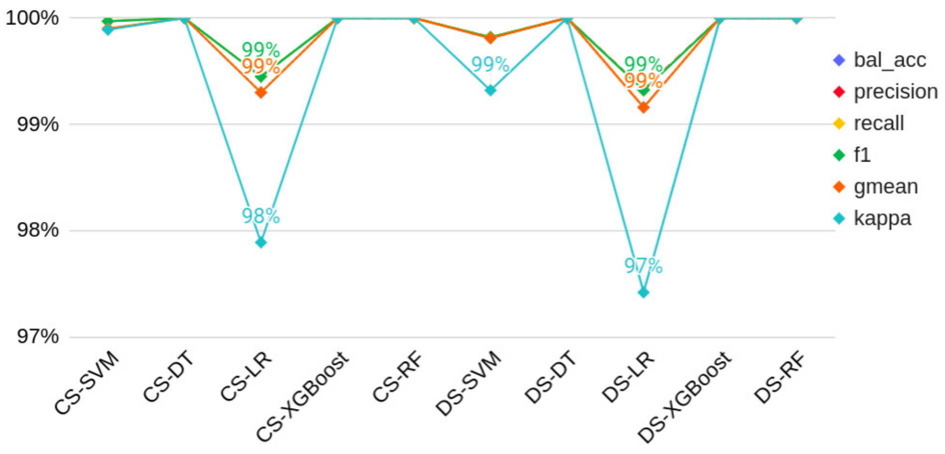

Figure 2: Comparison between Cost-sensitive(CS) and Data Sampling (DS) Methods for Training set.

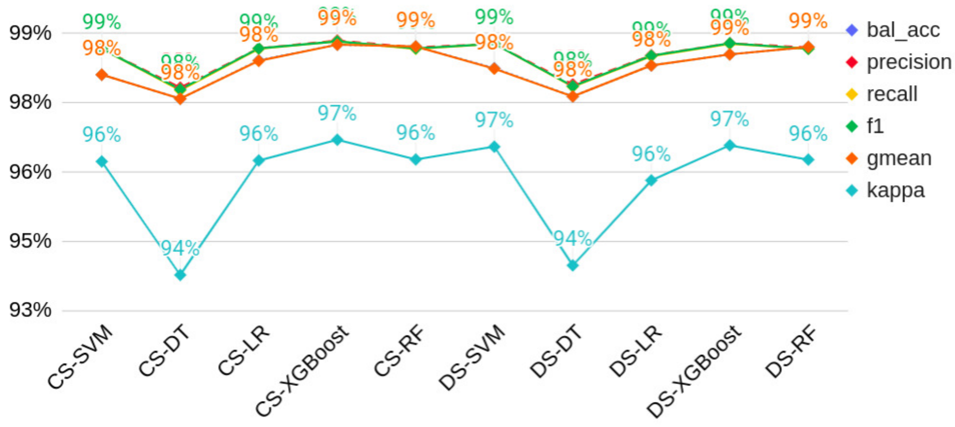

Figure 3: Comparison between Cost-sensitive(CS) and Data Sampling (DS) Methods for Validation set.

validation set, all cost-sensitive algorithms also achieved good results (above 97%), highlighting XGBoost, which obtained the highest scores, namely: balance accuracy equal 98.76%, precision equal 98.85%, recall equal 98.83%, f1-score equal 98.83%, g-mean equal 98.76% , and kappa equal 98.70%. The oversampling, undersampling, and the oversamplingundersampling combination methods achieved the same results in performance metrics, both in training and validation sets, c.f. we can note in Tables 5, 6, 7, 8, 9, and 10. In the training set, oversampling, undersampling, and the oversamplingundersampling combination methods achieved results (above 99%) in performance metrics. In the validation set, the three data sampling tested also achieved good results (above 97%), highlighting RF, which obtained the highest balance accuracy (98.71%) and g-mean (98.71%); and SVM and XGBoost, which obtained the highest precision (98.78%), recall (98.78%), f1-score(98.78%), and kappa (96.55%).

In the Figure 2 and 3 are showed the comparison of the metric performance results of cost-sensitive algorithms and of the data sampling methods, respectively. Both achieved good results and although the results of cost-sensitive algorithms were subtly higher than the data sampling methods results, statistical tests are needed to claim if there is a significant difference between them.

5 Conclusion

This paper investigated the distribution of classes in imbalanced classification problems and how it affects data preparation and modelling algorithms. We described the concept of cost-sensitive algorithms, which consider the imbalanced class distribution and aim to provide equitable treatment to both majority and minority classes. These algorithms enhance the accuracy of predictions on imbalanced datasets by capturing their intricacies. We presented a pipeline for configuring hyperparameters and training models to identify the best cost-sensitive algorithm for a specific study case. The cost-sensitive Extreme Gradient Boosting algorithm is highlighted as an effective solution for mitigating the impact of class imbalance and reducing misclassification costs in a lawsuit decision classification task using an imbalanced dataset.

References

Paula Branco, Luís Torgo, and Rita P. Ribeiro. 2015. A survey of predictive modelling under imbalanced distributions. *arxiv.org/abs/1505.01658*, abs/1505.01658.

Chao Chen, Andy Liaw, Leo Breiman, et al. 2004. Using random forest to learn imbalanced data. *University of California, Berkeley*, 110(1-12):24.

Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794.

Hong-Liang Dai. 2015. Class imbalance learning via a fuzzy total margin based support vector machine. *Applied Soft Computing*, 31:172–184.

Conselho Nacional de Justiça Departamento de Pesquisas Judiciárias. 2022 [Online]. Justiça em números 2022. Justiça em números 2022.

Charles Elkan. 2001. The foundations of cost-sensitive learning. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence - Volume 2*, page 973–978. Morgan Kaufmann Publishers Inc.

Mikel Galar, Alberto Fernandez, Edurne Barrenechea, Humberto Bustince, and Francisco Herrera. 2011. A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(4):463–484.

M.A. Hearst, S.T. Dumais, E. Osuna, J. Platt, and B. Scholkopf. 1998. Support vector machines. *IEEE Intelligent Systems and their Applications*, 13(4):18–28.

Max Kuhn, Kjell Johnson, et al. 2013. *Applied predictive modeling*, volume 26. Springer.

Yunqian Ma and Haibo He. 2013. *Imbalanced learning: foundations, algorithms, and applications*. John Wiley & Sons.

Joseph Berkson M.D. 1944. Application of the logistic function to bio-assay. *Journal of the American Statistical Association*, 39(227):357–365.

Shaoning Pang, Lei Zhu, Gang Chen, Abdolhossein Sarrafzadeh, Tao Ban, and Daisuke Inoue. 2013. Dynamic class imbalance learning for incremental lpsvm. *Neural Networks*, 44:87–100.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in python. *J. of Machine Learning Research*, 12:2825–2830.

J. Ross Quinlan. 1986. Induction of decision trees. *Machine learning*, 1(1):81–106.

Gerard Salton and Chris Buckley. 1988. Term-weighting approaches in automatic text retrieval. *Information processing & management*, 24(5):513–523.

Claude Sammut and Geoffrey I Webb. 2011. *Encyclopedia of machine learning*. Springer Science & Business Media.

Nguyen Thai-Nghe, Zeno Gantner, and Lars Schmidt-Thieme. 2010. Cost-sensitive learning methods for imbalanced data. In *The 2010 International joint conference on neural networks)*, pages 1–8. IEEE.

Xulei Yang, Qing Song, and Yue Wang. 2007. A weighted support vector machine for data classification. *International Journal of Pattern Recognition and Artificial Intelligence*, 21(05):961–976.

Zhuangyuan Zhao, Ping Zhong, and Yaohong Zhao. 2011. Learning svm with weighted maximum margin criterion for classification of imbalanced data. *Mathematical and Computer Modelling*, 54(3-4):1093–1099.