





Cross-validation Strategies for Balanced and Imbalanced Datasets

Thomas Fontanari^{1,2}, Tiago Comassetto Fróes¹,
and Mariana Recamonde-Mendoza^{1,2}

¹ Institute of Informatics, Universidade Federal do Rio Grande do Sul,
Porto Alegre, RS, Brazil

{[tvfontanari](mailto:tvfontanari@inf.ufrgs.br), [tiago.froes](mailto:tiago.froes@inf.ufrgs.br), [mrmendoza](mailto:mrmendoza@inf.ufrgs.br)}@inf.ufrgs.br

² Bioinformatics Core, Hospital de Clínicas de Porto Alegre, Porto Alegre, RS, Brazil

Abstract. Cross-validation (CV) is a widely used technique in machine learning pipelines. However, some of its drawbacks have been recognized in the last decades. In particular, CV may generate folds unrepresentative of the whole dataset, which led some works to propose methods that attempt to produce more distribution-balanced folds. In this work, we propose an adaptation of a cluster-based technique for cross-validation based on mini-batch k-means that is more computationally efficient. Furthermore, we compare our adaptation with other splitting strategies previously not compared and also analyze whether class imbalance may influence the quality of the estimators. Our results indicate that the more elaborate CV strategies show potential gains when a small number of folds is used, but stratified cross-validation is preferable for 10-fold CV or in imbalanced scenarios. Finally, our adaptation of the cluster-based splitter reduces its computational cost while retaining similar performance.

Keywords: Model evaluation · Cross-validation · Class imbalance

1 Introduction

Splitting a dataset into various subsets for training and validation is a fundamental part of machine learning and is present in multiple tasks, such as model evaluation, model comparison, and hyperparameter tuning. Some traditional methods to split datasets into training and test sets are holdout, bootstrap, and cross-validation (CV).

Although cross-validation is arguably the most popular partitioning method, it has some relevant drawbacks that have been studied in the last decades. Given

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001, and by grants from the Fundação de Amparo à Pesquisa do Estado do Rio Grande do Sul - FAPERGS [21/2551-0002052-0] and Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) [308075/2021-8].

its stochastic nature, CV may lead to poor estimates because of a partition-induced dataset shift [14], that is, some of the generated folds are not representative of the data. One generally handles this by using repeated cross-validation. However, since applying cross-validation is already computationally expensive, repeating it multiple times may be prohibitive.

The aforementioned issues are related to the randomized steps of CV. Therefore, a few methods have been proposed that attempt to improve the estimates of cross-validation by introducing a more deterministic process of generating folds. In one of the first works on the topic, Diamantidis *et al.* [9] introduced a clustering-based technique that relied on k-means (here referred to as CBD-SCV). However, k-means can be expensive when used as a splitting strategy for CV. At a similar time, Zeng *et al.* [24] proposed distribution-balanced stratified cross-validation (DBSCV), which was later adapted by Moreno-Torres *et al.* [14], introducing the distribution optimally balanced stratified cross-validation (DOB-SCV). Although DBSCV and DOBSCV have been compared before, there has been no direct comparison between them and the cluster-based methods.

In our work, we propose the use of mini-batch k-means as a way of reducing the computational cost of CBDSCV. Furthermore, we provide a comparison between CBDSCV, DOBSCV, and DBSCV, besides the traditional cross-validation techniques, on 20 datasets of various sizes, class imbalance levels, number of features, and number of classes. Our experiments aim to assess whether any cross-validation splitting strategy tends to outperform the others in terms of bias, variance, or computational cost.

The rest of the paper is structured as follows. In Sect. 2 we present the theoretical background of our work, followed by a description of our experiments in Sect. 3. Next, in Sect. 4, we present and discuss our results for balanced and imbalanced datasets. Section 5 revises other papers that presented efforts towards proposing cross-validation splitting strategies and were not directly compared in our experiments. Finally, Sect. 6 presents our conclusions and directions for future works.

2 k-fold Cross-validation Partitioning Methods

The traditional k-fold cross-validation (CV) [10] consists in dividing the given dataset into k folds. Each fold is then used once as the validation set, while the remaining $k - 1$ folds are used for training. Finally, the average performance obtained for each fold is the performance estimate of the k-fold CV. In general, k is set as 5 or 10, which makes it much more computationally tractable than leave-one-out cross-validation (LOOCV), besides showing less variance than LOOCV estimates. Furthermore, it is less biased than the holdout method, since it is able to use more instances for training than the holdout. K-fold cross-validation can also be used in a stratified fashion (k-fold SCV) to guarantee that the proportion of instances of each class is the same for all folds.

However, the instances assigned to each fold by traditional k-fold CV and SCV are selected randomly, which can cause some folds not to be good representatives of the whole dataset [14]. For instance, it is not guaranteed that all the

regions in the input space will be appropriately represented over all folds. This phenomenon may impact performance estimates and thus has been considered in various works (see Sect. 5 for references), leading to new splitting strategies based on the features of the data and not only on their class labels. The methods we have considered in this work are reviewed in the following sections, and we also describe the adaptation we developed.

2.1 Distribution-Balanced Stratified Cross-validation

Following Moreno-Torres *et al.* [14], the distribution-balanced stratified cross-validation (DBSCV) [24] attempts to generate folds representative of the full dataset by assigning neighboring instances to different folds. Specifically, DBSCV randomly selects an instance and assigns it to a fold; it then jumps to the nearest instance of the same class and assigns it to the next fold. These steps are repeated until all instances of that class have been assigned to a fold. The same process is applied to the other classes so that the folds have approximately the same number of instances per class. Assuming a balanced distribution of the instances, building pairwise distance matrices for each class has complexity $\mathcal{O}(C(\frac{N}{C})^2) = \mathcal{O}(\frac{N^2}{C})$, where N and C are the numbers of instances and classes. The search-and-hop step has complexity $\mathcal{O}(C\frac{N}{C}\frac{N}{C})$, so that the final complexity of the algorithm is $\mathcal{O}(\frac{N^2}{C})$.

2.2 Distribution Optimally Balanced Stratified Cross-validation

The distribution optimally balanced stratified cross-validation (DOBSCV) is a modification of DBSCV. It also starts on a random instance of the datasets, but instead of hopping to the closest one of the same class, DOBSCV finds the $(k-1)$ nearest neighbors of the current instance belonging to the same class and assigns each of them to a different fold. This process is repeated independently for each class, similarly to DBSCV, until all instances have been assigned to a fold. Our implementation of DOBSCV also uses a pairwise distance matrix for each class. Assuming balanced classes, building the matrices has complexity $\mathcal{O}(\frac{N^2}{C})$ and searching the k -NN for the selected instances in each class can be done in $\mathcal{O}(C\frac{N}{kC}\frac{kN}{C})$, resulting in an overall asymptotic complexity of $\mathcal{O}(\frac{N^2}{C})$.

2.3 Clustering-Based Approaches

Diamantidis *et al.* [9] introduced unsupervised stratification for cross-validation, based on dataset clustering. Although they have also explored hierarchical clustering, their main proposed algorithm is using k -means to cluster the dataset into M clusters. The instances inside each cluster are then sorted by their distances to the cluster center in ascending order. Finally, they assign adjacent instances to different folds, *i.e.*, they make a pass over the sorted list of instances assigning each to a different fold. Note that the number of folds K , clusters M , and

classes C need not be equal. We refer to this method as cluster-based stratified cross-validation (CBDSCV). The unsupervised stratification process, however, does not guarantee that the classes are stratified in the usual sense, *i.e.*, the method does not necessarily generate folds with the same proportion of instances per class as the original dataset. K-Means has an average complexity of $\mathcal{O}(MNT)$, where T is the number of iterations, and sorting each cluster can be done in $\mathcal{O}(MN \log N)$. Therefore, CBDSCV has a complexity given by $\mathcal{O}(MN(T + \log N))$.

Mini-Batch CBDSCV. The running time of the CBDSCV algorithm is generally dominated by k-means. Therefore, we propose the use of mini-batch k-means [21] as a way of reducing the cost of performing CBDSCV. Mini-batch k-means is an adaptation of k-means with two major differences. First, at each iteration, it selects only a batch of samples instead of the whole dataset. These samples are then assigned to the nearest centroid. Then, instead of computing the new centroid as the mean of all instances assigned to a cluster, it iterates over the instances of the cluster, updating the centroid at each instance using a learning rate η inversely proportional to the number of times this centroid has been updated previously. Mini-batch k-means converges faster than k-means while producing results that are only slightly worse [2, 21]. In the following sections, we will refer to our adaptation of CBDSCV as CBDSCV_Mini.

3 Experiments

The experiments performed here were designed to evaluate whether there is a cross-validation splitting strategy which generally outperforms the others in terms of bias, variance, or computational cost. Moreover, we also wish to study whether the imbalance of the datasets may influence the quality of the splitters estimations. Since the estimations they produce may depend on the dataset, classifier, and also on the metric being estimated, we experimented with 20 different datasets (from PMLB [16]) and 4 different classifiers. The datasets were selected so that two groups would be apparent, one with balanced and the other with imbalanced datasets. The complete list is shown in Table 1.

Note that we use the same class imbalance measure $I \in [0, 1]$ as in [16], defined by $I = \frac{K}{K-1} \sum_{i=1}^K \left(\frac{n_i}{N} - \frac{1}{K} \right)^2$, where K is the number of classes, n_i is the number of instances in class i , and N is the dataset size. Imbalance is 0 when the classes are equally distributed and approaches 1 when almost all instances belong to the same class. When analyzing balanced datasets, we evaluated the splitters in terms of their accuracy estimations, as this is the most common and traditional metric. However, when handling imbalanced datasets, we used the F1 score since accuracy is inappropriate in these cases. We used the average between the F1 scores computed for each class, *i.e.*, the macro average.

We chose learning algorithms that presented different biases and variance levels. Specifically, we experimented using Logistic Regression (LR), Decision Trees (DT), Support Vector Machines (SVC), and Random Forests (RF). We

Table 1. List of datasets used in the experiments. Datasets with Imbalance higher than 0.20 were considered imbalanced.

Dataset	Examples	Attributes	Classes	Imbalance	Clusters
allrep	3772	29	4	0.91	7
analcata_data_cyyoung8092	97	10	2	0.26	3
analcata_data_dmft	797	4	6	0.00	5
analcata_data_germangss	400	5	4	0.00	4
analcata_data_happiness	60	3	3	0.00	4
analcata_data_japansolvent	52	9	2	0.00	3
appendicitis	106	7	2	0.36	6
backache	180	32	2	0.52	5
car	1728	6	4	0.39	6
chess	3196	36	2	0.00	4
colic	368	22	2	0.07	5
dna	3186	180	3	0.08	1
flare	1066	10	2	0.43	5
hepatitis	155	19	2	0.34	5
movement_libras	360	90	15	0.00	5
new_thyroid	215	5	3	0.30	6
page_blocks	5473	10	5	0.76	4
postoperative_patient_data	88	8	2	0.21	4
vote	435	16	2	0.05	2
vowel	990	13	11	0.00	4

have used only the RBF kernel with SVC since linear decision functions could be represented by Logistic Regression. To avoid overfitting when handling class-imbalanced datasets, weights associated with the instances of each class during training were set to be inversely proportional to the class frequencies in the training set. Prior to the experiments, we tuned each classifier to each dataset using the entire data and grid search. The performance of each hyperparameter set was evaluated using 5-fold cross-validation and the hyperparameters which showed the highest F1 score were chosen. These selected hyperparameters for a classifier-dataset pair were fixed for the experiments so that the classifiers were always trained with the same hyperparameters independently of the splitting strategy being analyzed. With this approach, we aim to capture performance differences caused by variation in the splitting strategy rather than by variation in the hyperparameters values.

Finally, we compared three splitting strategies, CBDSCV, DBSCV, and DOBSCV, against traditional k-fold cross-validation and stratified k-fold cross-validation. We also included our adaptation of CBDSCV, which uses mini-batch k-means for faster computation of the clusters, using batches of size 100. Therefore, six different cross-validation splitting strategies were compared in terms of their bias and variance, as well as computational cost. Our implementations

of the splitting strategies, the selected hyperparameters, and the code used in experiments are available online¹.

3.1 Estimating the Bias and the Variance

The cross-validation methods considered here attempt to estimate the test performance of the learning algorithms fitted to the datasets. The bias of a cross-validation method is defined as the difference between the expected estimate and the (true) test performance [11]. Since we are working with real datasets, it is unfeasible to obtain the test performance. However, we can compute estimations for it using repeated holdout a large number of times, similarly to [3, 11]. Specifically, we estimated the true performance for each dataset and classifier by repeating a stratified holdout 100 times, using 90% of the dataset for training, and getting the mean value. We chose a small test set in order to reduce the bias caused by using smaller training sets, while we expect that the high number of repetitions will attenuate the variance of the holdout.

The expected estimate of each cross-validation technique was computed for each dataset by resampling 90% of the dataset without repetition 20 times and applying the cross-validation technique to obtain the estimates of the true performance. The average value of the 20 estimates was used as the expected estimate of the cross-validation method. That is, let CV_i be the performance estimate of running k-fold cross-validation on a given dataset and learning algorithm, with a chosen splitting strategy, then we approximated the expected cross-validation estimate as

$$\overline{CV} = \frac{1}{20} \sum_{i=1}^{20} CV_i. \quad (1)$$

Finally, we computed the bias using $b_{CV} = \overline{CV} - \hat{P}$, where \hat{P} is the estimation of the true performance that was computed using 100-times repeated stratified holdout, as described above.

The other important quantity that determines the quality of an estimator is its variance. We computed the variance of the cross-validation estimates using

$$s_{CV}^2 = \frac{1}{20-1} \sum_{i=1}^{20} (CV_i - \overline{CV})^2. \quad (2)$$

In this paper, however, we will work with the standard deviation (std) s , since we believe it is more easily readable. Note that an estimator with high variance may give poor results even if it has a low bias since one may not have the luck to obtain one of the estimates closer to the true value.

We evaluated the bias and variance of the six different dataset partitioning strategies over 20 different datasets and four classifiers. For each k-fold cross-validation strategy, we experimented with 2, 5, and 10 folds. Finally, we used accuracy and F1 as the performance metrics.

¹ <https://github.com/froestiago/K-Fold-Partitioning-Methods/tree/bracis22>.

3.2 Defining the Number of Clusters

The cluster-based method requires a number of clusters to be given as input. Ideally, we would compute the number of clusters right before each splitting is performed. However, this would be too computationally expensive, since the number of experiments performed is already large. Therefore, we have chosen to estimate the number of clusters for each dataset prior to the main experiments, and use this number (rounded to the nearest integer) for all cluster-based splitter methods. We have followed the same strategy as Diamantidis *et al.* [9] to estimate the number of clusters, which was based on repeatedly applying hierarchical clustering to small samples of the datasets and using a threshold on the similarity between clusters being merged to determine the number of clusters. The resulting number of clusters for each dataset is shown in Table 1.

4 Results and Discussion

The experiments performed as described in the previous section result in 80 different samples of bias and variance for each k -fold splitter, where $k = 2, 5,$ and 10 . Each of the 80 samples corresponds to a dataset-classifier pair. In the next sections, we describe the results grouped by balanced and imbalanced datasets in terms of class labels, resulting in 40 dataset-classifier pairs for each group. We focus mainly on the results with 2 and 10 folds, but the Figures for the 5-folds case are available in our git page (See footnote 1).

4.1 Balanced Datasets

The bias and standard deviations of each 10-fold cross-validation splitting strategy for all datasets and classifiers are summarized in Fig. 1. All the methods showed a general tendency to very low bias and similar standard deviations, indicating that there is no solution that consistently performs better than all others.

Note, however, that this does not imply that the accuracy (or F1) estimates produced by each partitioning strategy is not different. The p-values for the Friedman tests [7] comparing the estimates of the splitters are shown in Table 2. In particular, the p-value for the estimates considered here is 0.0279, suggesting that the bias estimates differ depending on the splitting strategy. However, there is no significant difference in terms of the standard deviations. Table 3 shows the number of times each method performed the best. For 10 folds, accuracy, and balanced datasets, stratified 10-fold CV had the most wins for both bias and std.

Reducing the number of folds increases the bias (in absolute terms) and the standard deviations, as shown in Fig. 2. However, the methods still have similar performance overall. We note, however, that DOBSCV and CBDSCV had an increase in the number of times they had the best results, while stratified CV showed worse results compared with its performance in the 10-folds scenario,

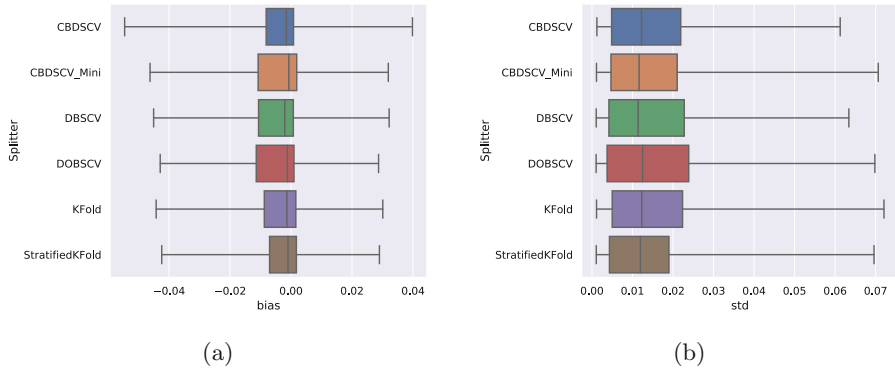


Fig. 1. (a) Bias and (b) standard deviation of each splitter method across all balanced datasets and classifiers. Each splitter runs 10-folds.

particularly with respect to the standard deviation of the estimates. This is an indication that the DOBSCV and CBDSCV can be useful when a reduced number of folds is desired so that the computational cost resulting from training various models can be reduced.

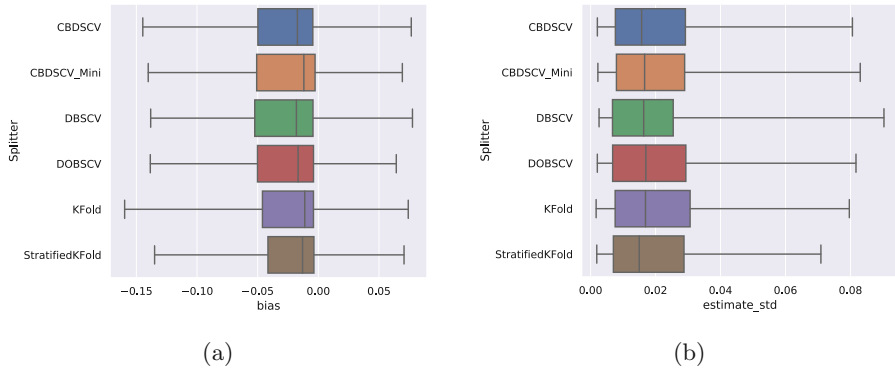


Fig. 2. (a) Bias and (b) standard deviation of each splitter method across all balanced datasets and classifiers. Each splitter runs 2-folds.

4.2 Imbalanced Datasets

The bias and standard deviations for 10-folds and imbalanced datasets are shown in Fig. 3. Since accuracy is not appropriate for studying imbalanced datasets, the bias and std were calculated for the f1-score estimations. Stratified 10-folds showed the best results for both bias and standard deviation. Tables 2 shows that the difference between the splitters is significant in the imbalanced cases, and

Table 2. p-values for the Friedman tests comparing whether the estimates produced by the splitters for each dataset-classifier pair differs. Smaller values mean that the hypothesis that the splitters produce similar estimates for the datasets and classifiers is unlikely. Values below 0.05 are in bold form.

Metric	Splits	Balance	p-value	
			bias	std
acc.	2	Balanced	0.11841	0.13278
		Imbalanced	0.45917	0.07770
	5	Balanced	0.58700	0.01481
		Imbalanced	0.09520	0.10409
	10	Balanced	0.02790	0.45271
		Imbalanced	0.72980	0.07762
f1	2	Balanced	0.01858	0.24038
		Imbalanced	0.10906	0.20719
	5	Balanced	0.00462	0.00055
		Imbalanced	<0.00001	0.00158
	10	Balanced	<0.00001	0.03294
		Imbalanced	<0.00001	0.00069

Table 3. Number of times each method had the best result in terms of bias or standard deviations, for various metrics, numbers of folds and dataset imbalance. The words *balanced* and *imbalanced* are abbreviated to *bal.* and *imb.*, respectively.

				CBDSCV	CBDSCV_Mini	DBSCV	DOBSCV	KFold	SKFold
acc	2	bal.	bias	2	9	3	12	3	11
			std	7	9	4	12	3	5
	5	bal.	bias	6	5	7	7	7	8
			std	8	8	8	5	2	9
	10	bal.	bias	4	6	7	7	5	11
			std	5	6	6	10	1	12
f1	2	bal.	bias	0	8	4	13	5	10
			std	8	8	5	11	3	5
		imb.	bias	6	5	9	9	5	6
			std	3	7	9	8	2	11
	5	bal.	bias	6	5	2	5	7	15
			std	11	8	5	5	1	10
		imb.	bias	6	6	5	1	7	15
			std	6	6	8	3	3	14
	10	bal.	bias	2	4	4	5	4	21
			std	5	8	5	9	2	11
		imb.	bias	4	4	2	1	5	24
			std	8	7	1	5	3	16

Table 3 shows that indeed stratified 10-fold presents the less biased and most consistent estimates for most datasets and classifiers. It is interesting to note that DBSCV and DOBSCV deal with class stratification by performing their splitting strategies per class. The fact that their performance was worse than stratified cross-validation suggests that there may be more appropriate ways to develop stratified versions of DBSCV and DOBSCV. The CBDSCV techniques, however, do not handle class imbalance directly.

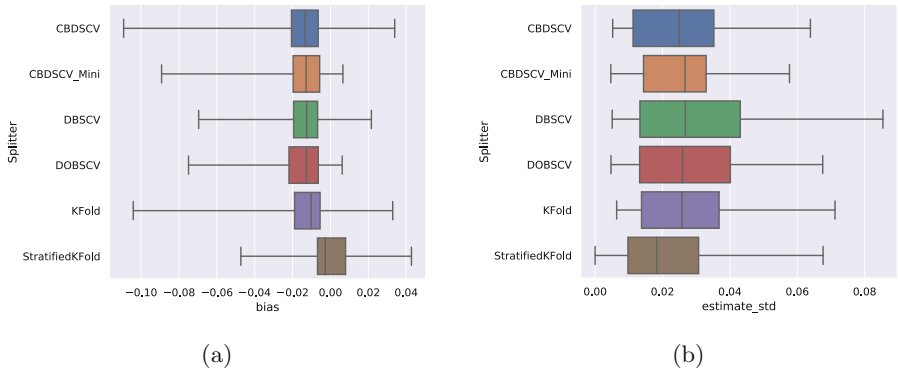


Fig. 3. (a) bias and (b) standard deviation of each splitter method across all imbalanced datasets and classifiers. Each splitter runs 10-folds for each splitter and the metric observed is the f1 score.

When one reduces the number of folds to 2, both the bias (absolute value) and the std of the estimates increase. More interestingly, the advantage that the stratified cross-validation had almost disappeared. This pattern is similar to the one observed for the balanced datasets. Table 3 shows that the number of times the SCV performs best indeed reduces when compared to the 10-folds case (Fig. 4).

4.3 Running Times

We also compared the running times of each splitting strategy. The running time of a k-fold splitting strategy for a dataset was obtained by averaging the running times of the splitting process over all the 20 runs and the 4 classifiers. We noticed that the running times obtained for 2, 5, and 10 folds were very similar, and therefore we consider only the 10-folds case in the following discussions. Figure 5 shows the running times of each splitter method for the 20 datasets considered. One can see that the classical strategies, KFold and StratifiedKFold, have negligible running times when compared to the others. Furthermore, we note that DBSCV and DOBSCV have higher variability depending on the

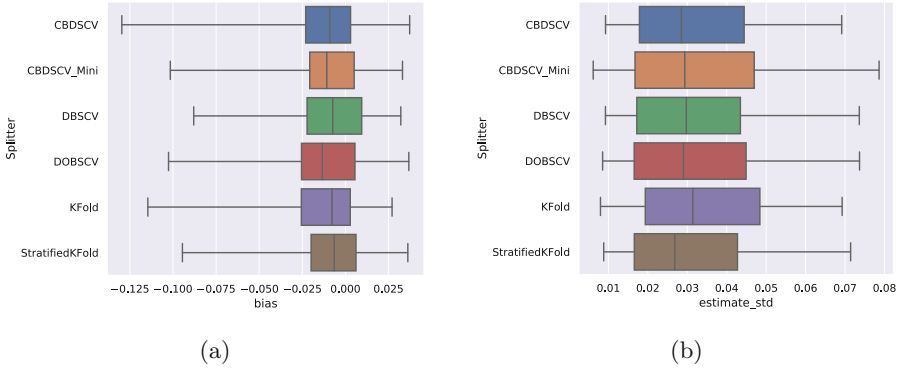


Fig. 4. (a) bias and (b) standard deviation of each splitter method across all imbalanced datasets and classifiers. Each splitter runs 2-folds for each splitter and the metric observed is the f1 score.

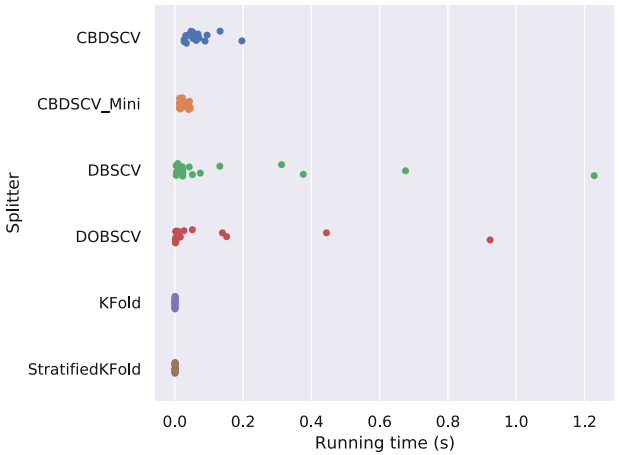


Fig. 5. Average running times in seconds across all the 20 datasets. The running times correspond to the use of 10 folds.

dataset, reaching the highest running times of all methods. In comparison, CBDSCV and CBDSCV_Mini have closer run times for all datasets.

Ignoring KFold and StratifiedKFold, however, DOBSCV was actually the fastest method in 14 out of 20 datasets, while CBDSCV_Mini was the quickest in the other six. Specifically, it was fastest in ‘analcata_data_germangss’, ‘movement_libras’, ‘analcata_data_dmft’, ‘appendicitis’, ‘page_blocks’, and ‘postoperative_patient_data’, which are the datasets with more instances. This is expected if one considers the algorithmic complexity of each method: DOBSCV scales quadratically with the number of samples.

4.4 Cluster-Based Splitters

When comparing only the cluster-based approaches, CBDSCV and CBDSCV_Mini, we observed that the running times on the minibatch version were smaller for all datasets, as expected. Specifically, it ran on average 2.4 times faster than CBDSCV. Furthermore, we have detected no significant change between estimations given by CBDSCV and CBDSCV_Mini. Table 4 shows the number of times each method performed better than the other, for all datasets and classifiers, and the p-value computed using the Wilcoxon test.

Table 4. Number of times each cluster-based method had the best result in terms of bias and variance. Only the cluster-based methods are considered here. The last columns shows the p-value for the two-sided Wilcoxon signed-rank test.

			CBDSCV	CBDSCV_Mini	p-value
Accuracy	2	bias	32	48	0.07208
		std	39	41	0.94265
	5	bias	43	37	0.88180
		std	40	40	0.91216
	10	bias	44	36	0.18399
		std	35	45	0.36078
f1	2	bias	35	45	0.04396
		std	38	42	0.64348
	5	bias	43	37	0.79371
		std	43	37	0.91596
	10	bias	33	47	0.12133
		std	38	42	0.61814

5 Related Work

Besides more recent theoretical works on traditional cross-validation estimates [4, 20, 22], various works have proposed cross-validation splitting strategies for different scenarios which are not directly related to our cases. Motl *et al.* [15] developed a technique based on linear programming for performing label-based stratification when the instances have more than one label at the same time, *i.e.*, multi-label datasets. Specific methods have also been developed for data drift situations, where some instances become obsolete over time [13], or for the specific case of dataset shift in credit card validation [19]. Cross-validation over graphs [8, 12] has also seen some recent works, as well as methods for reducing cross-validation computational cost in deep learning [6]. Cross-validation adaptations have been developed in order to handle duplicate data in medical records [1],

for calibration models in chemistry, [23] and for infrared and mass-spectroscopy images [17, 18]. All the methods cited above, however, are developed for different specific scenarios, whereas the methods explored here aim at tabular data and single-label datasets. Closer to the methods explored in this work are the proposals by Budka *et al.* [3] and Cervellera *et al.* [5]. In both works, the methods attempt to partition a dataset by generating samples whose distributions are as similar as possible to the distribution of the original data. We were not able to include them in this work due to lack of compatibility with the framework we had developed for our cross-validation methods comparison. Nevertheless, we will be including them in future works we are developing in this area of research. We note also that these two approaches have not been compared with each other yet.

6 Conclusion and Future Work

In this work, we proposed an adaptation of a cluster-based technique for splitting a dataset for cross-validation. We also compared various CV strategies using different classifiers for balanced and imbalanced datasets. We found that no method consistently outperforms all others in terms of bias or standard deviation when estimating accuracy using 10 folds and balanced datasets. In these cases, traditional stratified cross-validation remains a good choice. When the number of folds is reduced to 2, however, stratified cross-validation may produce accuracy estimates with higher variance than DOBSCV and the cluster-based techniques.

When considering F1 score estimates, traditional stratified cross-validation produced the best results in terms of bias and variance for most datasets and classifiers when used with 5 and 10 folds, for both balanced and imbalanced datasets. When the number of folds is reduced, however, F1 scores in balanced datasets may be better estimated by other methods such as DOBSCV and the cluster-based splitter. For imbalanced datasets, SCV remained the most frequent winner. In particular, traditional SCV was most significantly better when F1 score and imbalanced datasets were present. This suggests that better class-based stratification adaptations can be developed for DBSCV and DOBSCV. The development of a supervised version of CBDSCV is also an interesting topic for further work. Finally, we found no significant change in the quality of the estimations produced by CBDSCV and CBDSCV_Mini, whereas the mini-batch version is significantly less expensive in terms of computational cost.

We have not studied dataset characteristics such as the presence of subconcepts in the input space, as this kind of information is not easily extracted from a dataset. Those characteristics, however, may be relevant in determining which performance estimator should be used and may provide deeper insight into the use cases of each method. Similarly, we haven't analyzed deeper whether some splitting strategies may work better for each classifier or for datasets with different sample sizes. Finally, in future works, we intend to expand our experimental comparison and explore other approaches proposed in the literature [3, 5].

References

1. Bey, R., Goussault, R., Grolleau, F., Benchoufi, M., Porcher, R.: Fold-stratified cross-validation for unbiased and privacy-preserving federated learning. *J. Am. Med. Inform. Assoc.* **27**(8), 1244–1251 (2020). <https://doi.org/10.1093/jamia/ocaa096>
2. Bottou, L., Bengio, Y.: Convergence properties of the k-means algorithms. In: Tesauro, G., Touretzky, D., Leen, T. (eds.) *Advances in Neural Information Processing Systems*, vol. 7. MIT Press (1994)
3. Budka, M., Gabrys, B.: Density-preserving sampling: robust and efficient alternative to cross-validation for error estimation. *IEEE Trans. Neural Netw. Learn. Syst.* **24**(1), 22–34 (2013). <https://doi.org/10.1109/TNNLS.2012.2222925>
4. Celisse, A., Mary-Huard, T.: Theoretical analysis of cross-validation for estimating the risk of the k-nearest neighbor classifier. *J. Mach. Learn. Res.* **19**(1), 2373–2426 (2018). [JMLR.org](http://jmlr.org)
5. Cervellera, C., Maccio, D.: Distribution-preserving stratified sampling for learning problems. *IEEE Trans. Neural Netw. Learn. Syst.* 1–10 (2017). <https://doi.org/10.1109/TNNLS.2017.2706964>
6. Cheng, J., et al.: dwt-cv: dense weight transfer-based cross validation strategy for model selection in biomedical data analysis. *Futur. Gener. Comput. Syst.* **135**, 20–29 (2022). <https://doi.org/10.1016/j.future.2022.04.025>
7. Corder, G.W., Foreman, D.I.: *Nonparametric statistics for non-statisticians* (2011)
8. Dabbs, B., Junker, B.: Comparison of cross-validation methods for stochastic block models. Technical report, [arXiv:1605.03000](https://arxiv.org/abs/1605.03000), arXiv (May 2016), [arXiv:1605.03000](https://arxiv.org/abs/1605.03000) [stat] type: article
9. Diamantidis, N., Karlis, D., Giakoumakis, E.: Unsupervised stratification of cross-validation for accuracy estimation. *Artif. Intell.* **116**(1–2), 1–16 (2000). [https://doi.org/10.1016/S0004-3702\(99\)00094-6](https://doi.org/10.1016/S0004-3702(99)00094-6)
10. James, G., Witten, D., Hastie, T., Tibshirani, R.: *An Introduction to Statistical Learning*, vol. 112, chap. 5. Springer, New York (2013). <https://doi.org/10.1007/978-1-4614-7138-7>
11. Kohavi, R., others: A study of cross-validation and bootstrap for accuracy estimation and model selection. In: *IJCAI*, vol. 14, no. 2, pp. 1137–1145. Montreal, Canada (1995)
12. Li, T., Levina, E., Zhu, J.: Network cross-validation by edge sampling. *Biometrika* **107**(2), 257–276 (2020). <https://doi.org/10.1093/biomet/asaa006>
13. Maldonado, S., López, J., Iturriaga, A.: Out-of-time cross-validation strategies for classification in the presence of dataset shift. *Appl. Intell.* **52**(5), 5770–5783 (2021). <https://doi.org/10.1007/s10489-021-02735-2>
14. Moreno-Torres, J.G., Saez, J.A., Herrera, F.: Study on the impact of partition-induced dataset shift on k-fold cross-validation. *IEEE Trans. Neural Netw. Learn. Syst.* **23**(8), 1304–1312 (2012). <https://doi.org/10.1109/TNNLS.2012.2199516>
15. Motl, J., Kordík, P.: Stratified cross-validation on multiple columns. In: *2021 IEEE 33rd International Conference on Tools with Artificial Intelligence (ICTAI)*, pp. 26–31, November 2021. <https://doi.org/10.1109/ICTAI52525.2021.00012>
16. Olson, R.S., La Cava, W., Orzechowski, P., Urbanowicz, R.J., Moore, J.H.: PMLB: a large benchmark suite for machine learning evaluation and comparison. *BioData mining* **10**(1), 1–13 (2017)
17. Pérez-Guaita, D., Kuligowski, J., Lendl, B., Wood, B.R., Quintás, G.: Assessment of discriminant models in infrared imaging using constrained repeated random sampling-cross validation. *Analytica Chimica Acta* **1033**, 156–164 (2018). Elsevier

18. Pérez-Guaita, D., Quintás, G., Kuligowski, J.: Discriminant analysis and feature selection in mass spectrometry imaging using constrained repeated random sampling - Cross validation (CORRS-CV). *Anal. Chim. Acta* **1097**, 30–36 (2020). <https://doi.org/10.1016/j.aca.2019.10.039>
19. Qian, H., Wang, B., Ma, P., Peng, L., Gao, S., Song, Y.: Managing dataset shift by adversarial validation for credit scoring (2021). <https://doi.org/10.48550/ARXIV.2112.10078>
20. Santos, M.S., Soares, J.P., Abreu, P.H., Araujo, H., Santos, J.: Cross-Validation for Imbalanced Datasets: avoiding overoptimistic and overfitting approaches [research frontier]. *IEEE Comput. Intell. Mag.* **13**(4), 59–76 (2018). <https://doi.org/10.1109/MCI.2018.2866730>
21. Sculley, D.: Web-scale k-means clustering. In: Proceedings of the 19th International Conference on World Wide Web, pp. 1177–1178 (2010)
22. Wong, T.T., Yeh, P.Y.: Reliable accuracy estimates from k-fold cross validation. *IEEE Trans. Knowl. Data Eng.* **32**(8), 1586–1594 (2020). <https://doi.org/10.1109/TKDE.2019.2912815>
23. Xu, Q.S., Liang, Y.Z.: Monte Carlo cross validation. *Chemom. Intell. Lab. Syst.* **56**(1), 1–11 (2001). [https://doi.org/10.1016/S0169-7439\(00\)00122-2](https://doi.org/10.1016/S0169-7439(00)00122-2)
24. Zeng, X., Martinez, T.R.: Distribution-balanced stratified cross-validation for accuracy estimation. *J. Exp. Theor. Artif. Intell.* **12**(1), 1–12 (2000). <https://doi.org/10.1080/095281300146272>