machine learning & knowledge extraction

MDPI

Article

Cross-Validation Visualized: A Narrative Guide to Advanced Methods

Johannes Allgaier [1,2,*] and Rüdiger Pryss [1,2]

1 Institute of Medical Data Science, University Hospital Würzburg, 97080 Würzburg, Germany; ruediger.pryss@uni-wuerzburg.de
2 Institute of Clinical Epidemiology and Biometry, Julius-Maximilians-Universität Würzburg, 97080 Würzburg, Germany
* Correspondence: johannes.allgaier@uni-wuerzburg.de

Abstract: This study delves into the multifaceted nature of cross-validation (CV) techniques in machine learning model evaluation and selection, underscoring the challenge of choosing the most appropriate method due to the plethora of available variants. It aims to clarify and standardize terminology such as sets, groups, folds, and samples pivotal in the CV domain, and introduces an exhaustive compilation of advanced CV methods like leave-one-out, leave-*p*-out, Monte Carlo, grouped, stratified, and time-split CV within a hold-out CV framework. Through graphical representations, the paper enhances the comprehension of these methodologies, facilitating more informed decision making for practitioners. It further explores the synergy between different CV strategies and advocates for a unified approach to reporting model performance by consolidating essential metrics. The paper culminates in a comprehensive overview of the CV techniques discussed, illustrated with practical examples, offering valuable insights for both novice and experienced researchers in the field.

Keywords: train test split; cross-validation; grouped cross-validation; stratified cross-validation; time-based cross-validation; block cross-validation

check for updates

Citation: Allgaier, J.; Pryss, R. Cross-Validation Visualized: A Narrative Guide to Advanced Methods. *Mach. Learn. Knowl. Extr.* **2024**, *6*, 1378–1388. https://doi.org/10.3390/make6020065

Received: 11 March 2024
Revised: 13 May 2024
Accepted: 14 June 2024
Published: 20 June 2024

1. Introduction

In the ever-evolving field of machine learning (ML), the process of developing, deploying, and monitoring models is complex and multifaceted. A structured approach to navigating this complexity is provided by the Cross-Industry Standard Process for Data Mining (CRISP-DM) [1]. This methodology, developed in the late 1990s, has stood the test of time, offering a robust framework for executing ML projects. It comprises six phases: business understanding, data understanding, data preparation, modeling, evaluation, and deployment. Each phase plays a crucial role in ensuring the successful application of ML techniques to solve real-world problems.

A crucial aspect of the CRISP-DM cycle, particularly in the modeling and evaluation phases, is the practice of cross-validation. Cross-validation (CV) is a technique that helps to estimate the robustness and performance of models [2–4]. It is primarily used to predict how a model will perform in a production setting, an essential step given the potential variability of data in real-world scenarios. In CV, the data are divided into partitions, where the model is trained on one partition (we refer to this as D_{train}) and its performance is evaluated on a disjoint partition D_{test}. In this way, the robustness of the model can be assessed more accurately. This process is repeated multiple times, with different partitions, to reduce variability and ensure a more generalizable model.

In any project that follows the CRISP-DM cycle, there comes a point where an estimation of model performance and robustness is required to decide how to proceed with the project. A model is not robust if it performs poorly on new, unseen data. We refer to this phenomenon as overfitting [5,6]. Overfitted models usually have a low bias, but a high variance and perform poorly when predicting new data. Conversely, it may make sense to use simpler models and accept a higher bias in order to achieve better predictive power. The tradeoff between good performance and low bias is referred to as bias–variance tradeoff [7]. By cross-validating a model, we try to optimize the bias–variance tradeoff. By optimizing, we mean we want to minimize both bias and variance. Usually, if bias is high, variance is low and vice versa.

While examining the literature on machine learning, we observed a significant presence of homonyms and synonyms across the discourse related to cross-validation. To address this issue, this paper compiles a list of terms that are frequently used in connection with cross-validation. This glossary serves to increase clarity in the subsequent section, where we explain different cross-validation techniques and their possible combinations. In addition, we propose a formula that tries to balance the bias–variance tradeoff, addressing a crucial aspect of model evaluation. Further, we summarize all the methods discussed and provide a comprehensive assessment of the advantages and disadvantages of each approach. Finally, we discuss future directions of this research area where we point out that considering performance metrics of subject matter experts is equally important to gain value in ML projects.

Consider a machine learning (ML) or statistics project where your task is to train, evaluate, and deploy a model. Your dataset has groups and samples, and each sample has a timestamp, features, and a target (synonyms: outcome, dependent variable, label). To avoid confusion regarding homo- and synonyms, we provide the following list of definitions:

- Sample s (synonyms: instance, data point): Sample refers to a single unit of observation or record within a dataset.
- Dataset D: The total set of all samples available for training, validating, and testing ML models.
- Set: A batch of samples as a subset of the whole dataset D.
- Fold: A batch of samples as a subset of a Set. The term fold is often used in the context of k-fold CV.
- Group g (synonym: block [8]: A sub-collection of samples that share at least one common characteristic. Within the medical domain, for example, this characteristic could be a treating physician or a certain hospital. If multiple samples from the same patient are given, the patient him- or herself can be considered a group.
- Feature x (synonyms: predictor, input, or explanatory variable): An individual characteristic of the data that are given to the model for predicting the target. We denote the set of all features as X.
- Target y: (synonyms: outcome, dependent variable, or label): Within supervised learning, the property predicted by the model.

For further clarification, we illustrate the presented taxonomy concept on a small dataset in Figure 1.

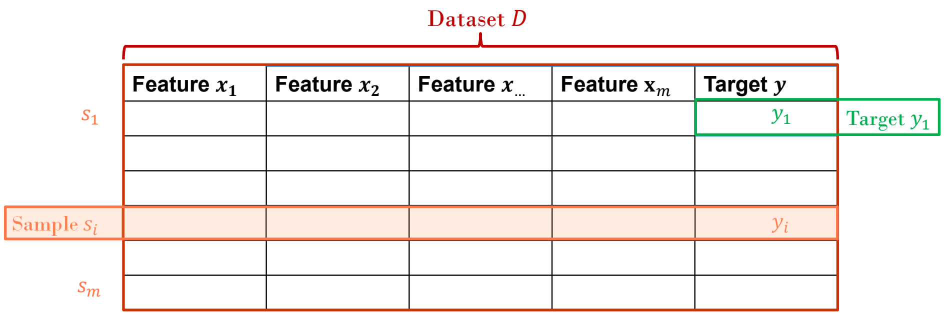


Figure 1. Illustration of the taxonomy concept. Please note that the sample s does not necessarily include the target y.

2. Overview of Cross-Validation Methods

In this section, various methods of CV are explained and visualized in more detail. By hold-out CV, we mean that we first split all available samples into two parts: D_{train} and

D_{test}. We perform cross-validation within D_{train} and finally test our chosen model in the hold-out D_{test} set. If you also cross-validate the hold-out set, we call this nested CV instead of hold-out CV. We further discuss possible combinations of CV approaches and conclude the section with a simple formula that can be used to target the bias–variance tradeoff and optimize the CV approach through a single number.

2.1. Hold-Out CV Methods

In all different CV approaches, we use a hold-out CV approach, which means we use the train set D_{train} to tune model parameters (this is where the CV actually happens) and the test set D_{test} to finally evaluate a chosen model. This idea goes back to [9]. When splitting the total data into training and test datasets, the size of the test set should be large enough to reflect the true distribution, but small enough to maintain the size of the training set. Common ratios are 80–20 or 70–30 for the split of training and test data. For large datasets, such as 10 million samples, a 99:1 train–test split may suffice, as 10,000 samples could adequately represent the target distribution in the test set. There is no one fits all approach, and the appropriate test size varies with the feature space complexity, use case, and target distribution, necessitating individual evaluation for each scenario.

The allocation of samples into folds and sets can happen at either group or sample level. By sample level, we mean that we do not correct for the distribution of the target (stratification) or assign batches of samples to separate groups during splitting. By group level, we mean that we keep groups separated from each other during validation and testing. To recall, a group is a sub-collection of samples that share a least one common characteristic.

2.1.1. Random Splitting of Samples: K-Fold CV

In previous research [2,4], the "classic" method to gauge model performance involves randomly splitting the dataset into k distinct folds, which is well known as k-fold CV. Here, $k-1$ folds constitute the training folds, while the remaining fold serves as the validation fold. Notably, the idea of randomly splitting samples was first mentioned in 1931 [10].

The term validation fold refers to the batch of samples that is used within the train set D_{train} to validate chosen model architectures (see Figure 2). This is not to be confused with a hold-out test set. A test set conclusively evaluates model performance. Depending on the use case you are working on, you might want to reserve a hold-out test set, not utilized for model validation until the end. When we use a validation fold within D_{train} and a hold-out test set D_{test}, we call this hold-out CV. Introducing a hold-out test set can help to mitigate overfitting to the dataset D, but it may reduce the available data for model training, potentially affecting the model's robustness. A hold-out CV approach with random splitting is depicted in Figure 2.

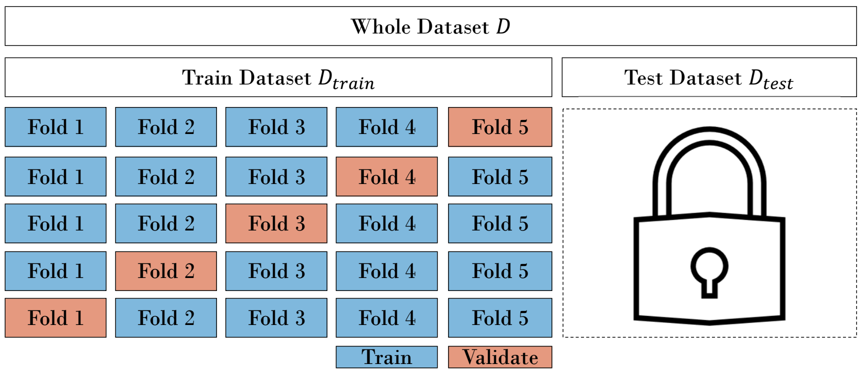


Figure 2. Common practice: Splitting the samples randomly [10], and at sample level into a train and test set. Then cross-validate within the train set using a k-fold CV (here, with $k = 5$) to tune model hyperparameters; then, validate these parameters on a validation fold. Finally, test the chosen model on a hold-out test set.

2.1.2. Common Variations of Cross-Validation with Random Splits

In the previous section, we have seen that we can randomly split the set D_{train} into k folds with each fold equal in size. In this section, we want to summarize three common variations of this approach. In these approaches, the data are not split on a fixed ratio, but on a certain number (1 or p). Note that in classic k-fold CV, each sample is used once. In these variations, however, samples can be used more than once or even not at all. We illustrate these approaches in Figure 3.

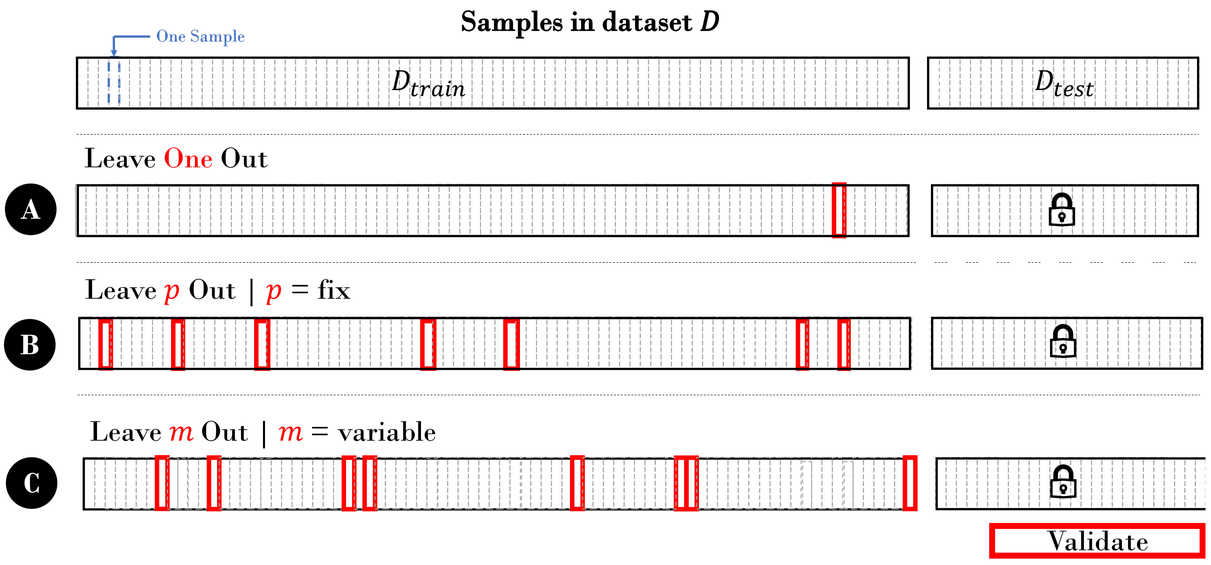


Figure 3. Cross-validation with left out samples to validate model performance within D_{train}. The number of validation samples depends on the method. In (**A**), only one sample is reserved for validation, whereas in (**B**), p samples are left out with $1 < p <$ number of samples in D_{train}, and in (**C**), the number of left out samples is random. For each variation of (**A**–**C**), there is the option to bootstrap samples which might lead to overlapping validation folds.

Leave-One-Out Cross-Validation

In this approach, all samples except one are used to train the model where the remaining sample is used to validate it. It is useful for smaller datasets where one wants to use as much data as possible to train a more robust model. However, it is also computationally expensive as one has as many validation folds as samples within D_{train}. The idea of leaving one sample out for validation of the model was firstly introduced by [9].

Leave-*P*-Out Cross-Validation

In leave-p-out CV, we leave more than one but less than n samples out to validate our model [11]. n is the number of samples in D_{train}. The difference between k-fold CV, leave-one-out CV, and leave-p-out CV lies in the number of samples included in the validation fold: In k-fold CV, each validation fold has $\frac{n}{k}$ samples. In leave-one-out CV, there is only one sample in the validation fold in each iteration, and in leave-p-out CV, you decide the size of the validation fold with p, which can be any value that is not necessarily $\frac{n}{k}$.

The choice of p in leave-p-out CV is a hyperparameter, and selecting an appropriate value for p involves considering the bias–variance tradeoff. Its choice depends on the size of your dataset, the complexity of your model, and the computational resources available. A smaller p leads to lower bias, more computational cost, and high variance. A larger p means lower variance, higher bias, and less data available in the training folds.

Monte Carlo Cross-Validation

The term Monte Carlo is often used in various fields of science and engineering to refer to methods that rely on random sampling. It is inspired by the Monte Carlo Casino in Monaco, known for its games of chance. The Monte Carlo CV (MCCV) approach has one key distinction to the other approaches: Here, the samples in the validation fold are chosen randomly with replacement [12]. That is, samples can overlap between folds. The approach goes back to [13]. Monte Carlo cross-validation (MCCV) is also referred to as Repeated random subsampling validation [14]. We visualize it in Figure 3C. Monte Carlo cross-validation introduces an additional layer of randomness compared to traditional *k*-fold cross-validation. In classic *k*-fold CV, the number of samples in each fold remains constant, and the data are randomly split. However, in Monte Carlo CV, not only are the data randomly partitioned, but the number of samples in each fold is also randomly determined. This introduces the possibility of including samples in zero folds or multiple times.

2.1.3. Stratification on the Target

Especially for medical data, it is not uncommon that the target distribution is strongly skewed in one direction, meaning there are many negative cases and only a few positive cases. In medicine, for example, subject matter experts refer to this as prevalence. Prevalence refers to the proportion or percentage of a specific condition or characteristic within a population. In our case, these are samples that refer to a minority class. When performing CV and random splitting, the smaller the batch size of the folds or set, the more likely it is to mitigate a complete class within this set or fold. This is illustrated in Figure 4. When using performance metrics like F1-score, precision, or recall, the observed model performance can significantly fluctuate across folds. This variability is not primarily due to overfitting the data but is rather a result of substantial variations in the target distribution between these folds. You have at least two options to avoid this: Bootstrapping, and using metrics that do not vary due to different class distributions between folds, such as the Area Under The Receiver Operating Characteristic (AUROC). Bootstrap stratification is a resampling technique that retains the original class distribution by generating bootstrap samples. For example, if your original distribution is 10:1 and one of your folds has a distribution of 5:1, you bootstrap the five samples to artificially create that 10:1 ratio.

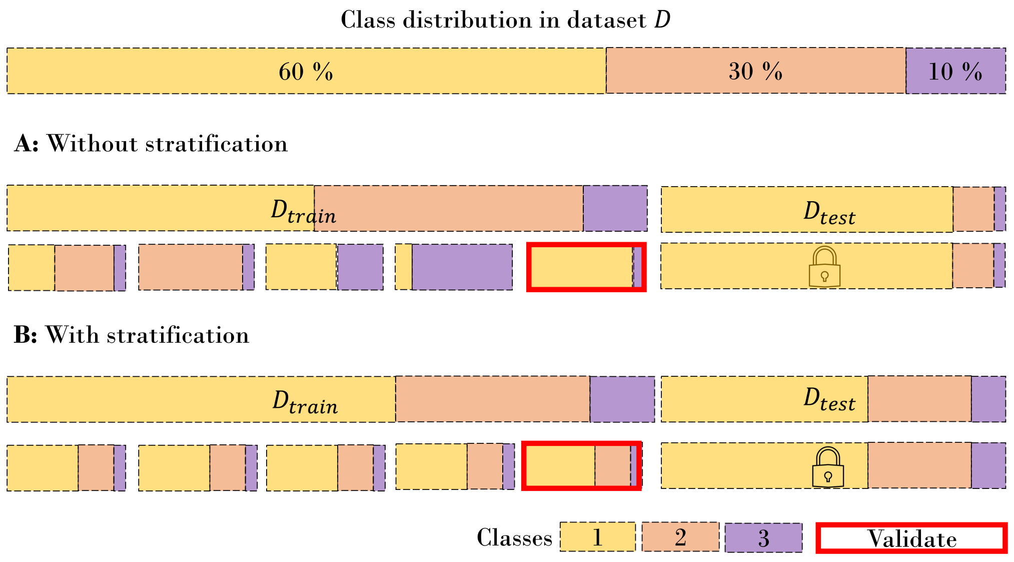


Figure 4. Stratification ensures the proportional representation of target variable distributions within subsets such as training sets or folds. (Scenario A) lacks stratification, risking underrepresentation of some classes in certain folds. In (scenario B), stratification preserves the overall class distribution across all folds.

2.1.4. Grouped Cross-Validation

Grouped or block CV involves aggregating samples into groups. While each group must appear in only one fold, one fold can have several groups. We illustrated this for four groups and one group per fold in Figure 5.

Separating groups helps to prevent overfitting on these groups [8]. One example within the healthcare domain are hospitals: If we group patient data by hospital and ensure that the hospitals are separated during training, the model is more likely to perform well on an unseen hospital after deployment. A broader set of examples may include spatial, hierarchical, and genetic structures [8]. If you have multiple groups that affect model performance, you might want to make this group a feature if that is feasible. For instance, hospital or physician could be a feature. However, if you want to deploy your model in another hospital, hospital cannot be a feature anymore.

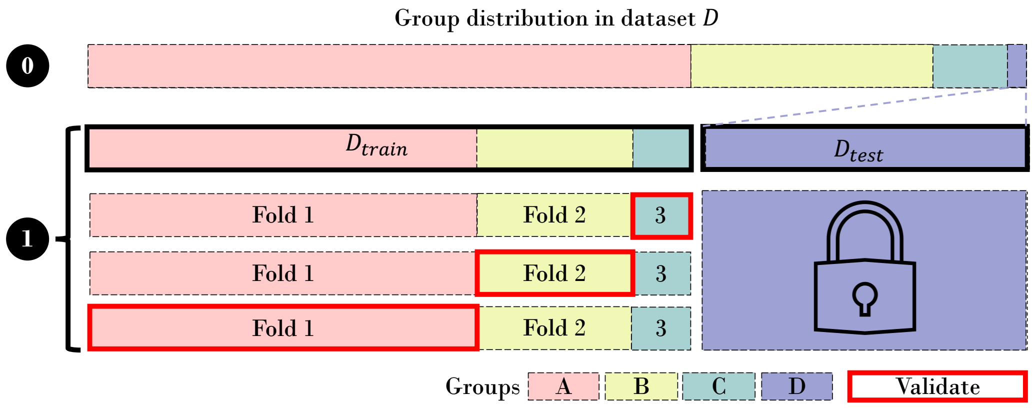


Figure 5. In 0, we can see the distribution of 4 groups that are present in dataset D. To prevent overfitting to a certain group, they should be separated from each other between sets. In 1, there is a grouped 3-fold CV with a hold-out set with only samples that belong to group D. Please note that we do not re-sample in D_{test}, the bar is just increased for illustration purposes.

2.1.5. Time-Split Cross-Validation

In the introduction, we defined a group as a set of samples that share a common characteristic. If we further know the timestamp of a sample, all samples from the same period of time (a day, a week, a season) can be considered a group [15]. Thus, time-split CV can be considered as a special case of grouped CV. Special for two reasons. First, if you want to train a time-series model, the groups should be ordered by time, and second, groups must not be shuffled during CV. In particular, the validation fold has to contain the "latest" data available at this time. Because we do not shuffle samples or groups, we can also state that this is not cross-validation, but just a train–validate split. However. we can think of four cases in time-split validation, which we drafted in Figure 6.

In scenario 1, the training dataset (=history) expands with each time step, while the validation dataset (=horizon) diminishes. This situation arises when there is a specific future time point of interest. In contrast, in scenario 2, the train data grow, while the size of the validation data remains constant. An example of this is dynamic system handling sensor data in real time with a constant prediction horizon. Scenario 3 mirrors scenario 1, with the exception being that the size of the historical data remains constant instead of growing. Lastly, in scenario 4, both sizes of historical and prospective datasets are fixed. These diverse approaches collectively fall under the umbrella term forward validation [16,17].

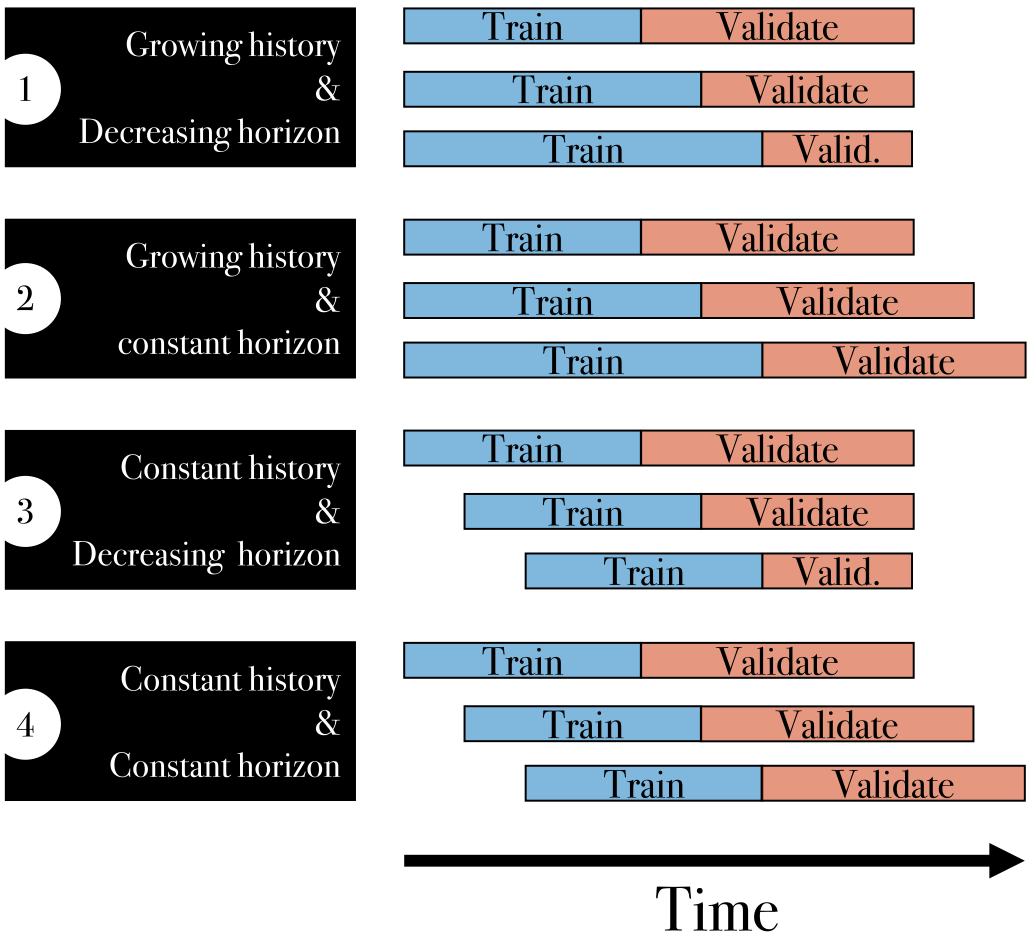


Figure 6. Time-split cross-validation with four variations: history growing and constant, and prediction horizon constant and decreasing.

Addressing Potential Concept and Data Drift in Time-Split Cross-Validation

Concept drift occurs when the statistical properties of the target variable, which the model is trying to predict, change over time [18]. Data drift, on the other hand, refers to changes in the input data features themselves while keeping the prediction task constant. Especially for longitudinal datasets, the datasets might contain drift. For instance, if the test set contains data that have drifted, given that the folds are stratified and groups are distinct, one can use the time-split approach to check whether the model is robust enough to still perform well in a drifted test set.

2.2. Limitations on Combining Advanced Methods

In classic k-fold CV, we split samples based on a ratio of D_{train}. For instance, a 5-fold CV would assign 20% of the samples to each fold. In contrast, leave-one-out, leave-p-out, and Monte Carlo assign a certain number of samples to the validation fold. When it comes to an extension or combination of the methods, however, we differentiate between the number of samples in the validation fold (1 for leave-one-out, >1 for classic k-fold, leave-p-out, and Monte Carlo). We visualize an overview of possible combinations in Figure 7.

	Combination		
Variation of CV	**Stratified**	**Grouped**	**Possible?**
Classic k-fold,		✓	Yes
Leave-P-Out, &	✓		Yes
Monte Carlo	✓	✓	It depends*
		✓	No
Leave-One-Out	✓		No
	✓	✓	No
		✓	It depends**
Time-Split	✓		Yes
	✓	✓	It depends**

Figure 7. Augmentation and combination options for CV methods. Leave-one-out has no combination options, whereas classic, leave-*p*-out, and Monte Carlo have. For leave-*p*-out, grouped CV works only if all groups in a fold contain at least *p* samples. * If you first group, then stratify, it might be possible to retain the class distribution of each group. If not, the group itself might be a valuable feature. If you first stratify, you might not be able to strictly separate groups. ** Grouped time-split validation works if all samples of a group fall within a time period. If not, this approach does not work. The checkmark means that stratification or grouping is applied to the variation of cv.

2.3. How to Target the Bias–Variance Trade Off

Up to now, we have mentioned the bias–variance tradeoff several times. In this short section, we offer a formula that might help to handle it. Given k validation folds within D_{train}, we can calculate k performance scores ν and average them. In a validation fold i, we call one performance score ν_i.

- $\bar{\nu}_{train} = \frac{1}{k}\sum_{i=1}^{k} \nu_i$
- To indicate the robustness of the model during training, we can also calculate the standard deviation σ as a function of ν:
  $\sigma(\nu_{train}) = \sqrt{\frac{1}{k}\sum_{i=1}^{k}(\nu_i - \bar{\nu}_{train})^2}$.
- Finally, we can determine the model performance in the test set, ν_{test}.

We now have three numbers, the average performance in the training set $\nu_{\bar{train}}$, the standard deviation of the model between folds during training, $\sigma(\nu_{train})$, and the test set performance ν_{test}. We now can either determine thresholds for two of three numbers and try to optimize the third one, or we combine the numbers to a final score ν_{final} and try to optimize this score. To suggest one combination: $\nu_{final} = \nu_{test} - \sigma(\nu_{train})$. Depending on the use case, one could also add a factor α to the standard deviation score. The more important the model robustness is, the higher α is:

$$\nu_{final} = \nu_{test} - \alpha\sigma(\nu_{train}) \tag{1}$$

It is crucial to be mindful of not overfitting our model to the test set D_{test} while fine-tuning for the bias–variance tradeoff. To avoid this, we have two options: limit how often we repeat this step, or introduce a third test set to evaluate the model's performance after deployment. This ensures the reliability and applicability of our proposed approach.

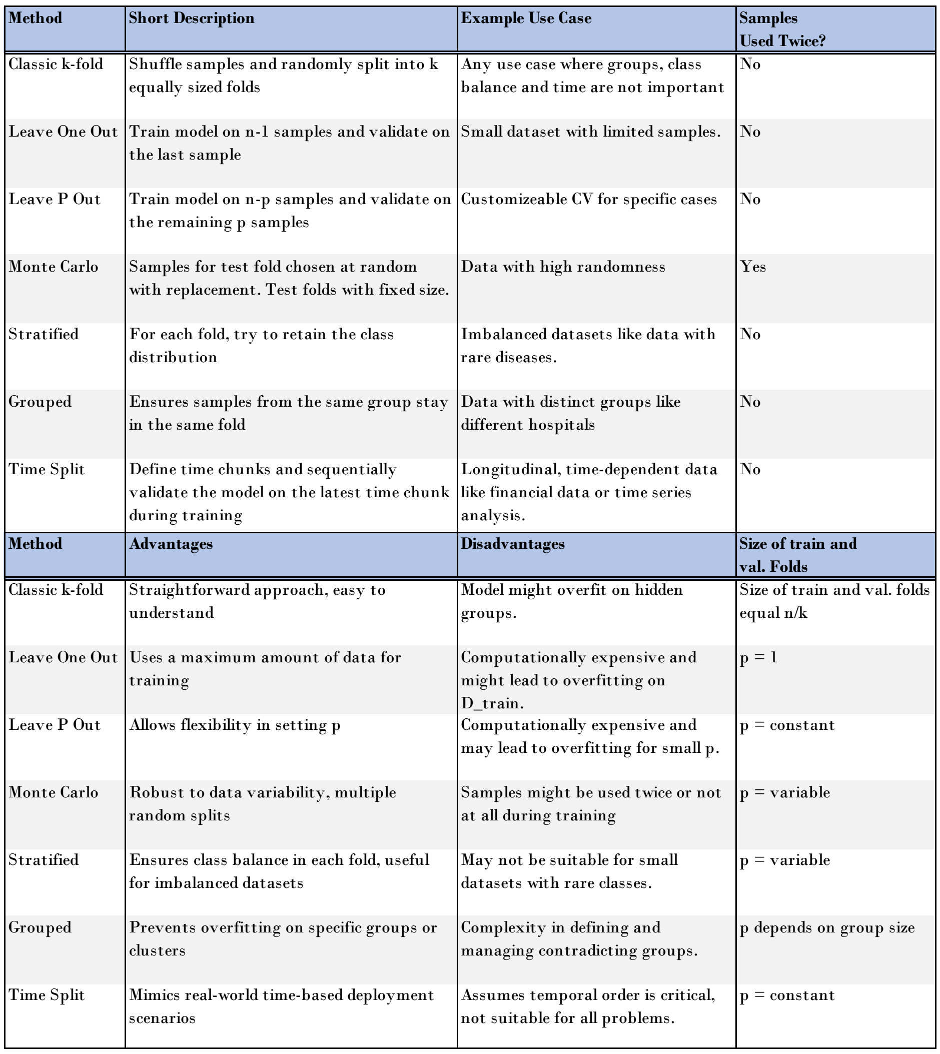
3. Discussion

In the introduction, we provided a short taxonomy on how to differentiate between datasets, fold, and sets. We differentiated between the sample and group level within CV to obtain more robust models and more accurate estimates of model performance. In the Methods Section, we provided several visualizations for advanced CV methods such as leave-one-out, leave-*p*-out, and Monte Carlo CV. We showed how stratification and separating groups can help to build more robust models. Next, we considered time-series

CV as a special case of grouped CV and showed that here the groups should not be shuffled and the number of training folds increases successively throughout the CV process.

Finally, we provided a template on how to report model performance and explained the options of optimizing either one metric while keeping the others above a threshold, or combining these numbers to a single one while controlling for model robustness via a hyperparameter α. We summarize all discussed methods with advantages, disadvantages, and example use cases in Figure 8.

Revisiting the Cross-Industry Standard Process for Data Mining (CRISP-DM) framework outlined in the introduction, it is imperative to acknowledge that selecting an appropriate cross-validation technique or a combination thereof represents merely a fractional component in the overarching schema required for the successful integration of a Machine Learning (ML) model within a production environment. Paramount considerations extend beyond mere methodology to encompass scalability, deployment architecture, and an array of non-technical parameters including legal compliance, transparency, trustworthiness, and the economic ramifications such as return on investment and enhancement of operational workflows, as highlighted by [19]. These dimensions gain clarity and can be more meticulously strategized when underpinned by a precise and reliable estimation of model performance, which is the focal point of this manuscript.



Method	**Short Description**	**Example Use Case**	**Samples Used Twice?**
Classic k-fold	Shuffle samples and randomly split into k equally sized folds	Any use case where groups, class balance and time are not important	No
Leave One Out	Train model on n-1 samples and validate on the last sample	Small dataset with limited samples.	No
Leave P Out	Train model on n-p samples and validate on the remaining p samples	Customizeable CV for specific cases	No
Monte Carlo	Samples for test fold chosen at random with replacement. Test folds with fixed size.	Data with high randomness	Yes
Stratified	For each fold, try to retain the class distribution	Imbalanced datasets like data with rare diseases.	No
Grouped	Ensures samples from the same group stay in the same fold	Data with distinct groups like different hospitals	No
Time Split	Define time chunks and sequentially validate the model on the latest time chunk during training	Longitudinal, time-dependent data like financial data or time series analysis.	No
Method	**Advantages**	**Disadvantages**	**Size of train and val. Folds**
Classic k-fold	Straightforward approach, easy to understand	Model might overfit on hidden groups.	Size of train and val. folds equal n/k
Leave One Out	Uses a maximum amount of data for training	Computationally expensive and might lead to overfitting on D_train.	p = 1
Leave P Out	Allows flexibility in setting p	Computationally expensive and may lead to overfitting for small p.	p = constant
Monte Carlo	Robust to data variability, multiple random splits	Samples might be used twice or not at all during training	p = variable
Stratified	Ensures class balance in each fold, useful for imbalanced datasets	May not be suitable for small datasets with rare classes.	p = variable
Grouped	Prevents overfitting on specific groups or clusters	Complexity in defining and managing contradicting groups.	p depends on group size
Time Split	Mimics real-world time-based deployment scenarios	Assumes temporal order is critical, not suitable for all problems.	p = constant

Figure 8. Summarizing all discussed cross-validation approaches. Generally speaking, the size of the train folds equals $\frac{n-p}{k}$ and in the test folds p.

4. Future Directions

Validating machine learning models is essential for assessing their resilience, as highlighted by [19] in their discussion on the challenges of machine learning validation. Our analysis has outlined various validation techniques and their possible integration, underscoring the necessity of evaluating a model's real-world applicability through its deployment performance. It is critical to acknowledge, as [20] emphasizes, that reliance on simulation-based testing is constrained by the underlying assumptions of the simulations, and thus cannot fully substitute for empirical testing in real-world scenarios. Consequently, we advocate for future studies to investigate the discrepancy between a model's anticipated performance in testing environments and its actual efficacy in operational contexts. When the objective of a machine learning (ML) project is to enhance value through task automation or workflow simplification, it is crucial to align ML performance metrics with domain-specific relevance. For example, in a healthcare setting where the ML system is tasked with classifying X-ray images, the key question is whether the system effectively identifies straightforward cases, thereby liberating human experts to devote more time to intricate scenarios. Considering this perspective, future research should explore the development of broader connections between domain-specific metrics and ML performance metrics. Specifically, the focus should not solely be on cross-validating through various data splits, but rather on adapting to diverse domain-specific contexts in each of the folds. For instance, treating distinct hospital workflows as unique domain-specific scenarios could be approached through grouped cross-validation. The novelty lies in the approach's orientation—seeing the ML use case through the subject matter expert's eyes and translating these needs into different cross-validated sub-use cases.

Author Contributions: Conceptualization, J.A.; methodology, J.A.; validation, J.A. and R.P.; resources, R.P.; writing—original draft preparation, J.A.; writing—review and editing, R.P.; visualization, J.A.; supervision, R.P.; project administration, R.P.; funding acquisition, R.P. All authors have read and agreed to the published version of the manuscript.

Funding: This publication was supported by the Open Access Publication Fund of the University of Würzburg.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: We did not code anything on data and thus neither provide data nor code.

Acknowledgments: Stephan Kolassa gave great feedback and valuable insights for this paper. We recommend checking out his account (https://stats.stackexchange.com/users/1352/stephan-kolassa (accessed on 15 Januaray 2024) on cross-validation and thank him for his collaboration.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Chapman, P.; Clinton, J.; Kerber, R.; Khabaza, T.; Reinartz, T.; Shearer, C.; Wirth, R. *CRISP-DM 1.0: Step-by-Step Data Mining Guide*; SPSS Inc.: Chicago, IL, USA, 2000; Volume 9, pp. 1–73.
2. Hastie, T.; Tibshirani, R.; Friedman, J. *The Elements of Statistical Learning*; Springer: Berlin/Heidelberg, Germany, 2009.
3. Hart, P.E.; Stork, D.G.; Duda, R.O. *Pattern Classification*; Wiley: Hoboken, NJ, USA, 2000.
4. Berrar, D. Cross-Validation. *Encycl. Bioinform. Comput. Biol.* ***2019***, 1, *542–545*.
5. Hawkins, D.M. The problem of overfitting. *J. Chem. Inf. Comput. Sci.* **2004**, *44*, 1–12. [CrossRef] [PubMed]
6. Ying, X. An overview of overfitting and its solutions. *J. Phys. Conf. Ser.* **2019**, *1168*, 022022. [CrossRef]
7. Belkin, M.; Hsu, D.; Ma, S.; Mandal, S. Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proc. Natl. Acad. Sci. USA* **2019**, *116*, 15849–15854. [CrossRef] [PubMed]
8. Roberts, D.R.; Bahn, V.; Ciuti, S.; Boyce, M.S.; Elith, J.; Guillera-Arroita, G.; Hauenstein, S.; Lahoz-Monfort, J.J.; Schröder, B.; Thuiller, W.; et al. Cross-validation strategies for data with temporal, spatial, hierarchical, or phylogenetic structure. *Ecography* **2017**, *40*, 913–929. [CrossRef]
9. Stone, M. Cross-Validatory Choice and Assessment of Statistical Predictions. *J. R. Stat. Soc. Ser. B* **1974**, *36*, 111–147. [CrossRef]

10. Larson, S.C. The shrinkage of the coefficient of multiple correlation. *J. Educ. Psychol.* **1931**, *22*, 45. [CrossRef]
11. Celisse, A.; Robin, S. Nonparametric density estimation by exact leave-p-out cross-validation. *Comput. Stat. Data Anal.* **2008**, *52*, 2350–2368. [CrossRef]
12. Xu, Q.S.; Liang, Y.Z. Monte Carlo cross validation. *Chemom. Intell. Lab. Syst.* **2001**, *56*, 1–11. [CrossRef]
13. Picard, R.R.; Cook, R.D. Cross-validation of regression models. *J. Am. Stat. Assoc.* **1984**, *79*, 575–583. [CrossRef]
14. Dubitzky, W.; Granzow, M.; Berrar, D.P. *Fundamentals of Data Mining in Genomics and Proteomics*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2007.
15. Bergmeir, C.; Benítez, J.M. On the use of cross-validation for time series predictor evaluation. *Inf. Sci.* **2012**, *191*, 192–213. [CrossRef]
16. Hjorth, U.; Hjort, U. Model selection and forward validation. *Scand. J. Stat.* **1982**, *9*, 95–105.
17. Hjorth, J.U. *Computer Intensive Statistical Methods: Validation, Model Selection, and Bootstrap*; CRC Press: Boca Raton, FL, USA, 1993.
18. Lu, J.; Liu, A.; Dong, F.; Gu, F.; Gama, J.; Zhang, G. Learning under concept drift: A review. *IEEE Trans. Knowl. Data Eng.* **2018**, *31*, 2346–2363. [CrossRef]
19. Baier, L.; Jöhren, F.; Seebacher, S. Challenges in the Deployment and Operation of Machine Learning in Practice. In Proceedings of the Twenty-Seventh European Conference on Information Systems (ECIS2019), Stockholm-Uppsala, Sweden, 8–14 June 2019; Volume 1.
20. Paleyes, A.; Urma, R.G.; Lawrence, N.D. Challenges in deploying machine learning: A survey of case studies. *ACM Comput. Surv.* **2022**, *55*, 114. [CrossRef]