A Reproduction Study and Evaluation of E-Fold Cross-Validation

Nick Petker
nick.petker@student.uni-siegen.de
University of Siegen
Siegen, Germany

ABSTRACT

This paper presents a reproduction of the e-fold cross-validation method proposed by Mahlich et al., which aims to reduce energy and resource costs by terminating cross-validation early once performance estimates stabilize. Using the original implementation and publicly available datasets from scikit-learn, we replicated their experimental setup and extended it with several parameter modifications to assess robustness and generalization. Our results confirm original findings: e-fold cross-validation achieves fold reduction, completing on average after 5.88 out of 10 folds, saving approximately 40% of resources. In 95.77% of all algorithm-dataset combinations, the performance estimates fell within a 95% confidence interval of a full 10-fold cross-validation. Performance differences remained low, with an average deviation of less than 2% for classification and less than 3% for regression. We further investigated the effects of modifying key parameters such as the maximum fold (e_{max}) and the stability counter. The proposed method is robust under default settings, but sensitive to specific changes.

KEYWORDS

machine learning, k-fold cross-validation, e-fold cross-validation

1 INTRODUCTION

1.1 Background

A well-established method for model evaluation and hyperparameter optimization in machine learning and related fields is k-fold cross-validation [5, 7]. In contrast to a simple hold-out validation, where the entire dataset is split once into a training/validation split, k-fold cross-validation partitions the dataset into k equal-sized folds. This kind of split ensures that each instance of the dataset is used both for training and evaluating. With k-fold cross-validation the performance of the model can be observed on the split data and the difference between bias and variance can be better understood [5]. The model is trained and validated k times, each time using a different fold as the validation set and the remaining folds for training. As a result of how the model is trained, it is supposed to improve performance compared to a single train/validation split [5, 6].

1.2 Research Problem

Although k-fold cross-validation offers its advantages in the robustness and supports better generalization, it comes also with its disadvantages: Since the model is trained k times, it requires k times more computational time, k more resources and k time more costs and energy consumption [7]. This also leads to increased CO_2 emissions, because the energy required for these additional computational time leads to the k times more CO_2 emissions. Another problem is that there is no optimal value for k [3, 7, 10]. The k value remains static across all datasets and algorithm in an experiment and while there is no optimal value there is a generally considered optimal, in literature and by most researchers, value between 5 and 10 [5, 6, 9]. The static value has the risk that the chosen value for k was either chosen to small, so that the model performance metric is not optimal, or it was chosen to large, so that the model performance metric is close to optimal but consumes more than needed resources.

1.3 Original Work

The concept of *e-fold cross-validation* was first proposed as a general idea in a preprint on OSF [4]. However, it was first implemented and evaluated in practice by Mahlich et al.[8], who applied the method systematically to classification and regression tasks. Furthermore, Baumgart et al.[1] also adapted the idea for recommender system evaluation. In the following, we focus on the implementation and evaluation by Mahlich et al., as their work provides the most comprehensive analysis.

Mahlich et al.[8] introduce *e-fold cross-validation*, a dynamic alternative to the traditional k-fold cross-validation. The key idea is to dynamically adjust the number of folds based on the performance stability, rather than using a static value of k in all evaluations. The motivation behind this approach is the reduce the computational time, resource usage and CO_2 emissions that come with the execution of all k-fold runs in an experiment.

The proposed method uses a stopping criteria based on the standard deviation of the performance scores of each fold. After every fold the algorithm checks whether the standard deviation has decreased or remained stable. If this happens in two consecutive iterations the evaluation process is stopped early. The process has two hyperparameter: e_{max} the maximum number of folds (here set to 10) and a stability counter limit (here set to 2). The stability counter ensures that the evaluation is not stopped too early but can still finish before reaching 10 folds.

To evaluate their approach, the authors conducted the experiments on 15 datasets (10 classification and 5 regression) available from the scikit-learn library, OpenML and kaggle. The classification datasets cover both binary and multi-class scenarios. Examples include the Mushroom, Diabetes Prediction and Iris datasets. For the regression datasets like the California Housing and CPU Small were used. In total 10 different machine learning algorithm were used, such as examples like Logistic Regression, Decision Trees, K-Nearest Neighbors, which all come from the scikit-learn library. The experiment were repeated 100 times for each dataset-algorithm combination.

For the classification problems, the authors used the F1-Score as the primary performance metric and as for the multi-class datasets the weighted variant. For regression Mean Absolute Error (MAE)

was used. The authors did not perform any hyperparameter optimization as the focus was on evaluating of their proposed method. The same goes for the algorithm as they were used in their default settings.

The results show that e-fold cross-validation terminated early in most cases, saving on average 4.33 folds per run which, if a fold counts as 10% resources, would be a reduction of roughly 40%. In 96% of all test runs the performance estimates obtained with the proposed method remained within the 95% confidence interval of the standard 10-fold results. For the binary classification the performance difference was typically below 1%, with outliers up to 5%, while for the multi-class classification and the regression typically below 2%, with outliers up to 16% for the multi-class classification and up to 12% for regression.

1.4 Research Goal

The goal of this work is to reproduce and evaluate the e-fold cross-validation method proposed by Mahlich et al. [8]. We aim to investigate whether the method can indeed reduce the number of folds needed and the resulting computational time and CO_2 emissions, without compromising model performance.

2 METHODOLOGY

In this section we describe the steps taken to reproduce the e-fold cross-validation approach proposed by Mahlich et al. [8]. We outline the replication of the original code, our experimental setup, which evaluation metrics were used and how our methology differs from the original study.

2.1 Replication

To ensure a correct understanding of the e-fold cross-validation approach, we began by reproducing the original results using the official implementation provided by Mahlich et al. [8]. The goal of this step was to verify that the method's stopping criterion and evaluation logic were implemented as described by the paper.

We executed the provided example script without any modification. The script include runs for a single classification dataset and compare the e-fold performance against the standard 10-fold cross-validation.

This replication step also helped us understand how the performance metrics were tracked, how the stopping condition based on the standard deviation was implemented and how the confidence intervals were computed. Based on this understanding, we adapted the code for our own experimental setup in the next steps.

Classification Algorithm	
Algorithm Name	**Source**
AdaBoost	scikit-learn
Decision Tree Classifier	scikit-learn
Gaussian Naive Bayes	scikit-learn
K-Nearest Neighbors	scikit-learn
Logistic Regression	scikit-learn
Random Tree Classifier	scikit-learn
Support Vector Classification	scikit-learn
MLP Classifier	scikit-learn
Regression Algorithm	
Decision Tree Regressor	scikit-learn
K-Nearest Neighbors Regressor	scikit-learn
Lasso	scikit-learn
Linear Regression	scikit-learn
Ridge	scikit-learn
Random Forest Regressor	scikit-learn
Support Vector Regression	scikit-learn
MLP Regressor	scikit-learn

Table 2: Supplementary information for all algorithms

2.2 Experimental Setup

Category	Component
CPU	AMD Ryzen 5 5600X (6 cores, 3.70GHz)
GPU	AMD Radeon RX 6650 XT
RAM	16GB DDR4
OS	Windows 10
Python	3.12.4
Libraries	scikit-learn, NumPy, SciPy, pandas, matplotlib

Table 1: Hardware specification

For the classification datasets we used: Breast cancer Wisconsin dataset, Iris plants dataset, Optical recognition of handwritten digits dataset and Wine recognition data. As for the regression dataset we used: fetch California housing and Diabetes dataset. All datasets were accessed via the scikit-learn library and performed without modification.

Table 2 shows the algorithms used for our experimental setup. All algorithms were applied with their default hyperparameter settings, as in the original study, to be able to compare the results.

All experiments were performed on a single local machine. The hardware specifications are listed in table 1. The code for our experiments are available on GitHub [11].

Overall we were able to reproduce the core results reported by Mahlich et al. The e-fold cross-validation method successfully stopped in most runs, typically after 4-7 folds. The observed performance metrics remained within the 95% confidence intervals of the standard 10-fold cross-validation, confirming the stability and validity of the method. Although minor differences in absolute

scores occurred possibly due to differences in dataset splits or library versions, the overall behavior of the approach matched the description in the original paper. Therefore we can conclude that the replication was successful.

2.3 Evaluation Metrics

In our evaluation, we primarily measure whether early stopping via the e-fold criterion produced comparable results to the full 10-fold cross-validation. The following metrics were recorded for each combination of algorithm and dataset across 100 randomized runs:

- **F1-Score**: Used for binary classification performance metrix
- **weighted F1-Score**: Used for multi-class classification performance metric
- **Mean Absolute Error (MAE)**: Used for regression performance metric
- **stopping fold (e)**: Each run we recorded the fold at which the early-stopping condition was triggered
- **Relative Performance Difference**: When an early stopping occured before $k = 10$, we calculated the percentage difference between e-fold and full 10-fold:

$$\text{Difference} = \frac{|N_{10} - N_e|}{N_{k10}} \times 100$$

This allowed us to see the difference between early stopping and accuracy

2.4 Differences from Original Study

In our reproduction study, we introduced several modifications to the original experimental setup to explore the robustness and applicability of the e-fold cross-validation.

Specifically we adjusted the number of folds k from 10 to 5 to evaluate the method under a different cross-validation. In another experiment the stability criterion counter was increased from 2 to 3 consecutive folds to require a stronger stability signal before stopping. To see the reliability of the results we increased the number of independent runs from 100 to 250 in another experiment.

We also tested the original implementation on 6 new algorithm: For classification: Random Tree Classifier, Support Vector Classification and MLP Classifier. As for regression: Random Forest Regressor, Support Vector Regression and MLP Regressor.

3 RESULTS & DISCUSSION

This section presents the results of our reproduction and analyzes the impact of our modifications to the original setup. We focus on the fold reduction, confidence intervals and the performance difference between the proposed e-fold cross-validation method and the 10-fold cross-validation and our modifications.

3.1 Fold Reduction

Through our experiments, without doing any modifications, e-fold cross-validation on average completed after 5.88 folds. The exact values of all algorithm datasets combination can be seen in table 3. That means that in this experiment e-fold cross-validation saved 4.12 folds on average compared to the full 10-fold cross-validation. If we value a single fold as 10% resources that means e-fold cross-validation saves about 40% of resources.

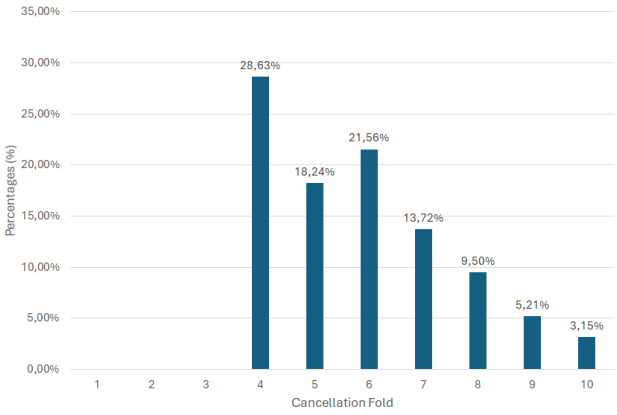

Figure 1: Distribution of early stopped folds

In figure 1 we see the distribution when the proposed method stopped early. The x-axis shows the number of folds where the method could terminate, while the y-axis represents the percentage frequency of theses values. The results show us that in about 28.63% of the time the method terminated after 4 folds. In about 3.15% of cases, early termination was not possible, resulting in the same number of folds as with 10-fold cross-validation.

We then started to modify the method in different kind of ways and were done separately. The first modification we do is to change the amount of runs. We do this to see if its limited to 100 runs or if its stable enough on more runs to see more outliner or performance differences. While we performed 7500 runs of combinations this way the outcome was comparable to the base version of the e-fold cross-validation. This result shows us that the amount of runs doesn't influence the proposed method.

The second modification we do is to change the k-value to 5. We do this to see how scalable the method is for smaller k-values. This results in two possible stopping points: fold 4 and fold 5. If stopped at fold 5 it results in the same number of fold when used with 5-fold cross-validation. This happened in about 44.51% of the time with the remaining 55.49% stopping at fold 4. On average e-fold cross-validation with e_{max} set to 5, it completed after 4.45 folds. If we again say that a single fold is the equivalent to 10% of resources, we did not even save 10% of resources this way. This result shows us that the proposed method does not work in saving resources in a small value of k.

The last modification we do is to change the stability counter from 2 to 3. This is intended to see how sensitive the method is to different kind of strictness for the performance. This changes the possible outcome of when e-fold cross-validation can terminate. The earliest stopping fold is now 5 instead of 4. This change leads to that, on average, the method completed after 7.27 folds. In our calculation with 1 fold equals 10% resources, we saved less than 30% of resources. The earliest stopping fold 5 terminated about 20.19% of the time while 10.32% it terminated at fold 10 meaning the same number of folds as with 10-fold cross-validation.

Model	Breast Cancer	Iris	Digits	Wine	Diabetes	California Housing	Average
Adaboost	5.85	5.90	6.06	5.89	*n/a*	*n/a*	5.93
Decision Tree Classifier	5.92	6.14	6.34	5.75	*n/a*	*n/a*	6.04
Guassian Naive Bays	6.14	5.65	6.20	5.47	*n/a*	*n/a*	5.87
K-Nearest Neighbor	5.77	5.29	5.91	5.99	*n/a*	*n/a*	5.74
Logistic Regression	5.72	5.42	5.85	5.56	*n/a*	*n/a*	5.64
Random Forest Classifier	6.18	5.69	6.11	5.27	*n/a*	*n/a*	5.81
Support Vector Classification	5.64	5.36	5.85	5.85	*n/a*	*n/a*	5.68
MLP Classifier	5.77	5.14	5.98	5.58	*n/a*	*n/a*	5.62
Decision Tree Regressor	*n/a*	*n/a*	*n/a*	*n/a*	5.96	6.09	6.03
K-Nearest Neighbor Regressor	*n/a*	*n/a*	*n/a*	*n/a*	5.99	6.01	6.00
Lasso	*n/a*	*n/a*	*n/a*	*n/a*	5.76	5.99	5.88
Linear Regression	*n/a*	*n/a*	*n/a*	*n/a*	6.23	6.11	6.17
Ridge Regression	*n/a*	*n/a*	*n/a*	*n/a*	5.82	6.11	5.97
Random Forest Regressor	*n/a*	*n/a*	*n/a*	*n/a*	6.03	5.87	5.95
Support Vector Regression	*n/a*	*n/a*	*n/a*	*n/a*	6.02	6.18	6.10
MLP Regressor	*n/a*	*n/a*	*n/a*	*n/a*	6.03	5.63	5.83
Average	5.87	5.57	6.04	5.67	5.98	6.00	**5.88**

Table 3: Average number of folds where e-fold cross-validation stopped

Configuration	Average Folds
Original Paper	5.67
Our Reproduction	5.88
Increased Runs (250)	5.85
$e_{max} = 5$	4.45
Stability Counter = 3	7.27

Table 4: Average number of folds for different configurations

Overall the original method shows consistent and meaningful fold reduction across a range of datasets and algorithms. However, our modifications in table 4 demonstrate that its efficiency depends on the chosen hyperparameter such as e_{max} and the stability counter. For practical use the selection of these parameters is necessary to ensure optimal resource savings without sacrificing evaluation quality.

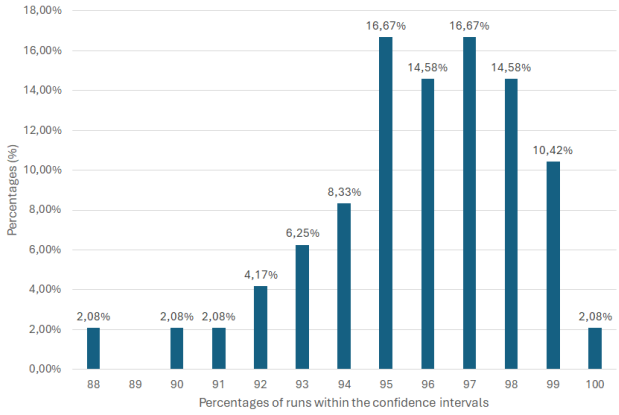

Figure 2: Distribution of number of iterations within the 95% confidence interval

3.2 Confidence Interval

While fold reduction demonstrates the potential efficiency gains of e-fold cross-validation, it is equally important to examine whether this comes at the cost of result reliability. Therefore, we now turn to an analysis of confidence intervals to assess the robustness of the method.

Without any modifications in our reproduction 95.77% of all algorithm-dataset combination the results were within a 95% confidence interval calculated using the t-distribution. This results in a total of 48 combination performing 100 runs for each algorithm on each dataset. The percentage of 95.77% was determined by analyzing the results of all iterations, with 4597 of 4800 falling within the confidence interval.

Figure 2 shows the distribution of number of iterations that fell within the confidence interval across all 48 different combinations. The x-axis shows the percentages of iterations that fall within the confidence interval, while the y-axis represents the percentage frequency of these values.

A accumulation is observed between 95% and 99% of all iterations. This shows that the method has a certain accuracy and reliability.

As seen in the fold reduction, increasing the amount of runs doesn't change much in the performance of e-fold cross-validation. The same can be seen in the amount of iterations, 7198 of 7500, that fall within the confidence interval with 250 runs. The percentage of 95.97% confirms the accuracy and reliability of the original implementation.

If we now change the e_{max} parameter to 5, we see that now only 1996 iterations of 3000 fall within the confidence interval. This results to a percentage value of 66.53%. This fall in accuracy reinforces that the e_{max} hyperparameter has to be chosen carefully.

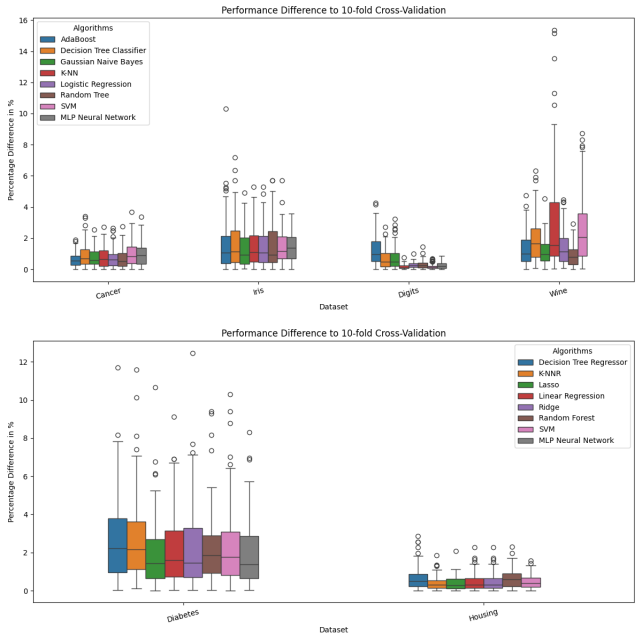

Figure 3: Percentage difference of performance metric between e-fold and 10-fold cross-validation

While changing e_{max} shows a drastic fall in accuracy, we don't see the same drastic fall when changing the stability counter to 3. We see that 2708 of 3000 iterations fall now within the confidence interval, resulting in 90.27%.

Overall, the original method demonstrates strong accuracy and reliability. However changing the parameter does have a strong impact of the accuracy. Especially when changing the e_{max} parameter the accuracy falls dramatically.

3.3 Performance Difference

After evaluating the potential efficiency and the statistical reliability of the method, we now tun to the performance metrics to assess the practical impact of e-fold cross-validation. We will see if there is a difference between e-fold cross-validation and 10-fold cross-validation in terms of performance.

We differentiate between the performance between classification and regression datasets. Results where e-fold cross-validation terminated with e = 10 are not included in the calculation, as the method does not terminate early at e = 10 and thus have the same score as 10-fold cross-validation.

The absolute percentage difference in performance metrics between e-fold cross-validation and 10-fold cross-validation remains on average low. For binary classification datasets it averaged less than 1% and for the multi-class classification dataset it averaged less than 2%. Regression Datasets averaged less than 3%. As seen in figure 3 there are outliers for all datasets. Binary classification datasets has outliers of up to 6% with multi-class dataset of up to 16%. Regression datasets have outliers of up to 13%.

We now investigate how the performance changes when modifying the parameters of the method. As we already seen with the fold reduction and the confidence interval changing the amount of runs doesn't influence the performance of the method. If we look at the averages after 250 runs we see similar results to the 100 runs.

While we have less information for k = 5, as we have fewer terminated runs, we can see that the performance difference to 10-fold cross-validation is still similar value wise. Meaning even if we change the e_{max} parameter, when we have a stopping fold, the performance difference to the 10-fold cross-validation is the same.

For the increase in the stability counter, we see on average less difference in performance. The binary and multi-class classification datasets are both less than 1% and the regression datasets are now less than 2% differences on average.

Overall, the results show that e-fold cross-validation achieves comparable performance to standard 10-fold cross-validation. While there are still some outlier and the comparison was only done, when the method terminated before e = 10, it shows its robustness. Our modifications further confirm this robustness but with a low e_{max} the method can occasionally become less reliable as half of the time there is no early stopped fold available.

3.4 Discussion

Our reproduction confirms that the default configuration of e-fold cross-validation is a robust and efficient method for reducing resources without significantly compromising performance or reliability.

The method also proved to be stable across different random seeds and datasets variations, and increasing the amount of runs had no significant impact on the results. However our experiments also show that the performance and reliability of the method depend on the correct choice of hyperparameters. Reducing the maximum number of folds (e_{max}) led to a noticeable drop in confidence interval coverage highlighting a trade-off between early stopping and robustness. While increasing the stability counter made the method more conservative and slightly improved result consistency.

Overall e-fold cross-validation has an interesting concept to try to save resources in comparison to 10-fold cross-validation. Nevertheless careful tuning of the method's parameters is necessary to ensure the desired balance between efficiency and accuracy. While our experiments were done on a relatively small limited size, we believe that the method may have limited scalability across different configurations.

4 LIMITATIONS

While our reproduction aims to faithfully reproduce and extend the e-fold cross-validation approach, there are several limitations worth noting:

- **Dataset Selection**: We did not use all datasets from the original study as some could not be faithfully located.
- **Model Hyperparameter Tuning**: We applied default hyperparameter settings for all algorithms, in line with the original study. We did not perform any hyperparameter optimization, which may limit the best performance results.
- **Hardware Constraints**: All experiments were conducted on a single local machine. Thus we did not investigate how much difference the impact of different configurations has.

- **Energy Consumption**: We did not exactly measure how much energy the system used overall during the experiments.

5 ACKNOWLEDGMENTS

This work was conducted as part of the Machine Learning Praktikum at the University of Siegen [2].

REFERENCES

[1] Moritz Baumgart, Lukas Wegmeth, Tobias Vente, and Joeran Beel. 2024. e-Fold Cross-Validation for Recommender-System Evaluation. arXiv:2412.01011 [cs.LG] https://arxiv.org/abs/2412.01011

[2] Joeran Beel and Lukas Wegmeth. 2025. Machine Learning Praktikum. *Universisät Siegen* (2025).

[3] Urwah Imran, Asim Waris, Maham Nayab, and Uzma Shafiq. 2023. Examining the Impact of Different K Values on the Performance of Multiple Algorithms in K-Fold Cross-Validation. In *2023 3rd International Conference on Digital Futures and Transformative Technologies (ICoDT2)*. IEEE, to appear in conference proceedings, 1–4. https://doi.org/10.1109/ICoDT259378

[4] Beel J., Wegmeth L., and Vente T. (2024, June 6). e-fold cross-validation: A computing and energy-efficient alternative to k-fold cross-validation with adaptive folds. ((2024, June 6)). https://doi.org/10.31219/osf.io/exw3j

[5] Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani, and Jonathan Taylor. 2023. *An Introduction to Statistical Learning: with Applications in Python*. Springer, Cham. https://doi.org/10.1007/978-3-031-38747-0

[6] Ron Kohavi. 1995. A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI'95)*, Vol. 2. Morgan Kaufmann, Montreal, Quebec, Canada, 1137–1143. San Francisco, CA, USA.

[7] Osval Antonio Montesinos López, Abelardo Montesinos López, and José Crossa. 2022. *Multivariate Statistical Machine Learning Methods for Genomic Prediction*. Springer, Cham. https://doi.org/10.1007/978-3-030-89010-0

[8] Christopher Mahlich, Tobias Vente, and Joeran Beel. 2025. From theory to practice: implementing and evaluating e-fold cross-validation. In *International Conference on Artificial Intelligence and Machine Learning Research (CAIMLR 2024)*, Jun Liu, Ankush Ghosh, Chee Wei Tan, and Haiquan Zhao (Eds.), Vol. 13635. International Society for Optics and Photonics, SPIE, 1363505. https://doi.org/10.1117/12.3058568

[9] Bruce G. Marcot and Anca M. Hanea. 2021. What is an optimal value of k in k-fold cross-validation in discrete Bayesian network analysis? *Computational Statistics* 36, 3 (2021), 2009–2031. https://doi.org/10.1007/s00180-020-00999-9

[10] Isaac Nti, Owusu Nyarko-Boateng, and Justice Aning. 2021. Performance of Machine Learning Algorithms with Different K Values in K-fold Cross-Validation. *International Journal of Information Technology and Computer Science* 13, 6 (2021), 61–71. https://doi.org/10.5815/ijitcs.2021.06.05

[11] Nick Petker. 2025. Open Source Reproduction of e-fold cross-validation. https://github.com/ShadNick/Evaluation-of-e-Fold-Cross-Validation.