University of Siegen

Intelligent Systems Group
Bachelor of Science

Energiebewusste k-fache Kreuzvalidierung

Energy-Aware k-fold Cross-Validation

Submitted by:	Christopher Mahlich
Study programme:	Computer science
Matriculation number:	1620221
Supervisor:	Prof. Dr.-Ing. Joeran Beel
Advisor:	Tobias Vente

23. September 2024

Abstract

In this thesis, a novel alternative to k-fold cross-validation in machine learning is presented, which is called e-fold cross-validation. E-fold cross-validation aims to be an energy-efficient variant of k-fold cross-validation. E-fold cross-validation dynamically adjusts the number of folds and consequently reduces computational resources while maintaining accurate performance estimates. This dynamic adjustment is implemented by a stopping criterion which checks after each fold whether the standard deviation of the previously evaluated folds has repeatedly decreased or has not changed significantly. Once the stopping criterion is met, the e-fold cross-validation is terminated early. E-fold cross-validation was evaluated on 15 datasets and 10 machine-learning algorithms. Compared to 10-fold cross-validation, e-fold cross-validation needed about 4 folds less for comparable results, resulting in savings of about 40% evaluation time, computational resources and energy consumption. The percentage difference in performance metrics between e-fold and standard 10-fold cross-validation was generally less than 2%. Statistical significance was confirmed with 96% of iterations falling within the confidence interval for all algorithms tested.

In hyperparameter optimization, the optimization with e-fold cross-validation was able to identify the same best hyperparameter combination as the optimization with 10-fold cross-validation in 55% of the cases. However, e-fold cross-validation needs, on average, 4.6 folds less.

In addition, the e-fold cross-validation method can stop even earlier by adjusting its own four hyperparameters and, therefore, save even more resources. On the other hand, the hyperparameters can also be adjusted in a way that e-fold cross-validation stops later and thus achieves better performance.

The results show that e-fold cross-validation is a robust and efficient alternative to the conventional k-fold cross-validation providing reliable performance evaluation at lower computational costs.

Preface and Acknowledgements

This bachelor thesis was written as part of my Computer Science studies at the University of Siegen. The choice of the topic, "Energy-Aware k-fold Cross-Validation", resulted from my strong interest in the field of machine learning, which was awakened in the module "Introduction to Machine Learning" during my studies.

Together with my supervisor, Prof. Dr.-Ing. Joeran Beel, I developed the research question for this bachelor thesis. Thanks to his expertise in the field of machine learning, I gained valuable insights that went beyond what I had learned during my studies. Together we also developed an accompanying paper, which will be presented at the 2024 International Conference on Artificial Intelligence and Machine Learning Research (CAIMLR).

I would like to thank Prof. Dr.-Ing. Joeran Beel for his support throughout the entire process, as well as my advisor Tobias Vente. Both have contributed to the further development of this work with their suggestions and feedback. The collaboration was not only instructive for me, but also a lot of fun.

Special thanks go to my family, who supported me during my studies.

Christopher Mahlich

Plettenberg, 21.09.2024

Contents

List of Figures

List of Tables

List of Algorithms

1 Introduction

K-fold cross-validation is a well-established method for model evaluation and hyperparameter optimization in machine learning and related fields [10][23]. Compared to a hold-out split, where the entire dataset is split into a single train/validation split, k-fold cross-validation ensures that each instance of the dataset is used for both training and validation. With k-fold cross-validation, the performance of the model can be observed on different subsets of the data, and the trade-off between bias and variance can be better understood [10]. As a result, models are trained and validated k times with different subsets, which is supposed to improve the performance and variability of the model compared to a single train/validation split and therefore is expected to provide more reliable results [10][16]. Hence k-fold cross-validation is usually preferred over a simple train/validation split [10][49].

While k-fold cross-validation offers advantages, it comes with disadvantages that are, however, generally accepted: k-fold cross-validation requires k times more evaluation time, k times more computational resources and consequently, k times higher costs and energy consumption [23]. The increased energy consumption also results in a higher level of CO_2 emissions, because the energy required for the additional computations is directly linked to the number of emissions generated resulting in k times more CO_2 emissions. Furthermore, there is no fixed, optimal value for k that is always ideal [25][9][23]. In literature and by most researchers, a k between 5 and 10 is considered optimal [16][21][10]. This k remains typically static across all datasets and algorithms during one experiment [23]. This has a risk that k was not chosen large enough and thus the model performance metric could not be optimal. On the other hand, there is a risk that k was chosen unnecessarily large, so that the model performance metric is close to the optimum, but more resources are consumed than necessary. To illustrate the risk of a too-high k value with an example: When comparing the performance metric of two algorithms and after 4 folds it is already clear that algorithm A is significantly outperforming algorithm B, additional folds would probably be an unnecessary use of resources, as the likely superior algorithm has already become clear.

The goal of this thesis is to develop a dynamic and energy-efficient alternative to k-fold cross-validation, called e-fold cross-validation. The difference to the conventional k-fold cross-validation is that e-fold cross-validation dynamically adjusts the number of folds based on the stability of the previously validated folds. In contrast to the fixed number of validations in k-fold cross-validation, e-fold cross-validation introduces a dynamic stopping criterion, which checks after each fold e whether the standard deviation of the previous folds remains stable. If no more significant changes occur, the method is stopped early after e folds.

The idea is that e is chosen intelligently and individually for each situation, ensuring that it is as small as possible to saves resources, but still large enough to achieve results that are close to the optimal results of the standard k-fold cross-validation with $k = 10$.

The following three key research questions will be analyzed to achieve the goal of this thesis and to examine whether e-fold cross-validation is a suitable energy-efficient alternative to k-fold cross-validation. First, how does e-fold cross-validation perform in direct comparison with 10-fold cross-validation? Second, is e-fold cross-validation suitable for hyperparameter optimization? Third, is it possible to further optimize e-fold cross-validation?

In the following Section 2 the existing research on a fixed k in k-fold cross-validation, a dynamic k in k-fold cross-validation and energy saving in machine learning is analyzed. Then, in Section 3, the e-fold cross-validation is described in detail and the algorithm is explained. The methodology Section 4 explains the experimental setup used to obtain the results. Section 5 describes and analyses the results of the work. Finally, in Section 6, the thesis ends with a summary and a conclusion.

2 Related Work

The existing research can be categorized into three main groups. First, determining the optimal size of k in k-fold cross-validation. Second, alternative approaches to a fixed k, including dynamic methods. Third, studies that focus on saving energy in machine learning.

There are several studies dealing with the question of the optimal k value in k-fold cross-validation. An often recommended value is $k = 10$, because it balances bias and variance. This recommendation goes back to Kohavi, who could show in an empirical study that $k = 10$ is the best choice in many cases because k values that are too small increase the bias and k values that are too large increase the computational effort without offering significant advantages in model performance [16]. This observation is supported by several studies.
Marcot et al. conducted a comprehensive study on the optimal choice of the k value in the context of discrete Bayesian networks. They developed several variants of Bayesian networks with different statistical properties and tested them using different values for k (2, 5, 10, 20, n-5, n-2, n-1) on datasets of different sizes. Their results show that $k = 10$ is a good choice in most cases, especially for smaller datasets. For larger datasets $k = 5$ is sufficient to optimize computation time without significantly affecting model performance [21]. Since the study was only conducted with discrete Bayesian networks, the results cannot be generalized to other algorithms.
Verma et al. [44] also confirm that k = 10 provides the best results for Decision Trees, which supports the widely used recommendation by Kohavi [16]. They investigated the impact of different values for k on the performance of a Decision Tree algorithm. In detail, they tested k values of 3, 5, 7 and 10. Verma et al. suggest that k = 10 is a good choice, especially for small to medium datasets, as it balances overfitting and computational effort [44]. The work of Verma et al. provides a good understanding of the application of k-fold cross-validation to Decision Tree models. However, in comparison to other studies such as that by Oyedele [26] or Marcot et al. [21], their study tests not many values for k and not enough datasets to make a generalizable statement.
Nti et al. also recommend $k = 10$ in their work, but they point out that in some cases $k = 7$ can provide better results. In their study, they analyzed different values for k (3, 5, 7, 10, 15, 20) and applied different machine learning algorithms, including Gradient Boosting Machine, Logistic Regression, Decision Tree and K-Nearest Neighbors [25]. However, in their study, they only used one dataset, which limits the generalisability of the results. Although $k = 7$ appeared good for the algorithms tested, it remains questionable whether this result is transferable to other datasets and algorithms.
Oyedele extends this discussion with an experimental approach to determine the optimal k for k-fold cross-validation for neural networks. In the study, the k values are tested in the range from 2 to 1000. The results show that the best results were obtained with k values between 10 and 20 [26]. The study also shows that very large k values underestimate the model performance, which may be

an indication that a too large value for k could bias the performance of the model [26]. This supports the recommendation to choose k in the range of 10 to 20 to achieve a good balance between estimation accuracy and computational effort, especially for neural networks. Oyedele's results thus confirm the general tendency in the literature that $k = 10$ is often a good choice [26]. In the same way as Marcot et al. [21] and Verma et al [44], it is important to note that these studies only tested one specific algorithm and it is, therefore, difficult to make a general statement about the choice of k.

In addition, studies such as those by Imran et al. [9] and Karal [13] suggest that there is no universally optimal k value. Both studies showed that the choice of k strongly depends on the algorithm used and the underlying data [9][13]. Imran et al. analyzed in detail k values between 2 and 20 for different machine learning algorithms and could not determine the one best k value that is optimal for every algorithm [9]. In his work, Karal also investigated the effects of different k values on support vector machines (SVMs) with different kernel functions. He showed that no individual k value consistently delivered the best results, but that the optimal choice of k varied depending on the dataset and model [13]. His results underline the need for flexible, dynamic approaches to optimize the k value depending on the particular application.

Wong et al. investigated whether repeated k-fold cross-validation provides more reliable estimates of accuracy. They found that the estimates from different replicates are highly correlated, especially with a large number of folds, which can lead to an underestimation of variance. The results of their study suggest that a higher k value for k-fold cross-validation with a small number of replicates is preferable to a smaller k value for multiple replicates to obtain more reliable estimates of accuracy performance [48]. Thus, they emphasize the importance of developing new dynamic methods that are more resource efficient, as higher k values lead to higher resource consumption. Wong et al. provide a valuable analysis of the dependencies between repetitions of k-fold cross-validation, but the results of this study largely relate to one algorithm. Therefore, it is difficult to generalize the results [48].

In summary, the studies show that k = 10 is often considered the standard for k-fold cross-validation. Nevertheless, a data-specific adjustment is advantageous in some cases, depending on the algorithm used, the dataset and the computing resources available. Higher k values often offer no significant advantages, especially when repeated cross-validation or larger datasets are used, so a dynamic choice of the k value would be useful.

One of the dynamic methods is the racing method. Thornton et al. introduced the racing method to eliminate bad performing model configurations at an early stage [38]. This method, also known as F-Race, uses statistical tests to evaluate the performance of models step-by-step and dynamically adjusts the number of folds used in the k-fold cross-validation. Models that perform badly in the first few folds can be eliminated at an early stage without having to analyze all k folds. This reduces the number of evaluations required [38].
Bergman et al. analyzed the possibility of early stopping of cross-validation in automated machine learning and found that by stopping cross-validation early, the model selection process can be speeded up without compromising the performance of the models [5]. This study investigated two different approaches for the early stopping of k-fold cross-validation. An aggressive approach and a forgiving approach. The aggressive approach terminates the cross-validation of a model if the average score after the evaluation of the first folds is worse than the best average score so far. The forgiving approach, on the other hand, only stops the k-fold cross-validation when the current average score of a model is worse than the worst individual fold of the best model [5]. In comparison to e-fold cross-validation, the stopping criteria in this study are based on the currently achieved average value of the evaluated hyperparameter combination and not on the standard deviation. The goal of the study also differs from the e-fold cross-validation. Here, the goal is to explore more hyperparameter combinations in the same amount of time to achieve better overall performance. In contrast, e-fold cross-validation aims to save computational resources by reducing the number of folds.
Soper introduced another dynamic alternative, Greedy k-fold cross-validation, which has the goal of speeding up the model selection process by optimizing the cross-validation process itself [36]. The Greedy approach dynamically adjusts the number of folds needed. In contrast to the traditional k-fold cross-validation, where all folds of a model are evaluated one after the other, the greedy approach decides adaptively which fold is evaluated next. The selection of the next fold is based on the model's previous performance. If a model already performs badly in the first folds, it is eliminated at an early stage without having to go through all the remaining folds. In this way, the algorithm focuses on the more promising models, saving resources and eliminating bad models at an early stage [36]. Greedy cross-validation differs from e-fold cross-validation in that it instantly eliminates models that perform poorly in early folds so that the weak models are not computed any further. In contrast, e-fold cross-validation dynamically adjusts the number of folds by evaluating the stability of model performance to save resources while still ensuring reliable performance evaluation.
In addition to this Greedy approach, Sorper has also introduced a combination of Greedy cross-validation with the Successive Halving method [11], known as Greedy Successive Halving [35]. This method combines the advantages of both approaches. While Greedy cross-validation dynamically adjusts the number of folds, Successive Halving ensures that weaker models are eliminated in several rounds by first testing them with smaller datasets and then further evaluating only the most promising models with increasingly larger datasets [11][36][35].

In this way, the number of folds to be calculated and the amount of training data are optimized step by step. Sorper describes this combination as an extremely effective method for speeding up model selection and hyperparameter optimization because it takes advantage of both methods [35].
Anguita et al. developed a dynamic approach for k-fold cross-validation that was specifically applied to support vector machines. Instead of using a fixed value for k, such as 5 or 10, their method uses k as a hyperparameter and dynamically adjusts it during model selection [2]. Here, k can assume any value in the range from 3 to the number of data points, whereby several different splits of the dataset are tested. With this dynamic choice of k and the consideration of different data splits, the best k for the given problem can be found [2]. However, in contrast to e-fold cross-validation and the dynamic methods described so far, the focus of this method is on improving the robustness of the error estimation rather than speeding up the process or saving resources. Consequently, it has a higher resource consumption due to the execution of multiple hyperparameter combinations for k.
E-fold cross-validation is an extension of the previously mentioned methods. In contrast to these methods, e-fold cross-validation uses an intelligent selection of folds based on statistical stability and is not based on checking whether the aggregated average score is better than the previous one. Furthermore, the e-fold cross-validation method has the primary goal of saving resources and not finding the best model performance. Therefore, badly performing models are not eliminated as they are in other methods. The dynamic methods described focus more on finding the best model combination in minimal time and less on achieving the goal with fewer resources. Nevertheless, they offer a good approach such as evaluating the folds step by step as it is also used in e-fold cross-validation.

Recent research has highlighted a critical problem in the machine learning community: Training and deploying machine learning models, especially deep learning algorithms, requires significant computational resources, resulting in substantial energy consumption and high CO_2 emissions [37][30][42]. This underlines the urgent need to develop energy-efficient methods to reduce the growing environmental impact of machine learning research.
In the field of recommender systems, Vente and Wegmeth et al. have shown how much energy is consumed by recommender systems and how much CO_2 is emitted. Their findings reveal that energy consumption is very high, with deep learning models, in particular, consuming an enormous amount of energy [42].
Similarly, Patterson et al. have shown through comprehensive studies that the energy consumption of machine learning, especially during training, is also notably high [27]. They provide concrete examples, such as the energy consumption of training large models like GPT-3, which consumes significant amounts of CO_2 [27]. In addition, they presented four key best practices that can drastically reduce energy consumption and CO_2 emissions. These four best practices are using more efficient model architectures, specialized hardware, cloud computing and selecting data center locations with cleaner energy sources [27].
Building on this, Lacoste et al. take a more targeted approach to quantify

the carbon emissions of machine learning by introducing the Machine Learning Emissions Calculator [17]. This Machine Learning Emissions Calculator estimates the environmental impact of training machine learning models by considering key variables such as server location, local energy grid, and hardware type [17].

Based on the work presented, the awareness of energy consumption is increasing and is being addressed in more studies, which is confirmed in the work of Verdecchia et al [43]. They comprehensively analyzed the current state of research in the field of Green AI and concluded that this field of research has grown considerably since 2020 [43]. They identified that the focus is on monitoring energy consumption, hyperparameter tuning, benchmarking models and implementing AI models [43]. Furthermore, they found that the majority of Green AI strategies achieve significant energy savings. However, they emphasize the urgency of conducting research in such a way that the developed methods are easily reproducible so that they can be easily integrated into real-world applications [43].

This again shows the importance of developing energy-saving machine learning methods that can be applied in practice. Roth et al. have put this into practice in their work [29]. Their work is aimed to investigate and optimize active approaches to reduce resource consumption in neural networks. They present techniques that minimize computational effort, memory requirements and energy consumption while maintaining prediction accuracy as far as possible [29]. The work demonstrates through experiments that methods such as quantization, network pruning and structural efficiency can significantly reduce resource consumption. These techniques make it possible to use neural networks in resource-limited environments such as embedded systems or energy-efficient devices [29].

Building on the growing awareness of energy consumption in machine learning, this thesis has the goal to develop a more energy-efficient alternative to the traditional k-fold cross-validation. It aims to significantly reduce the computational overhead by optimizing the number of folds without compromising the performance of model. This thesis is in line with existing Green AI research that aims to reduce energy consumption in machine learning. In particular, it addresses the need highlighted by Verdecchia et al. [43] for energy-efficient hyperparameter optimization and model evaluation. In addition, the method is easy to implement and can therefore be quickly integrated into existing real-world applications and is not based on theoretical foundations only.

3 E-fold cross-validation

In this section, the e-fold cross-validation method is described and defined. This is the first implementation of this method, an idea that was recently introduced [3]. E-fold cross-validation is a modified method of k-fold cross-validation. E-fold cross-validation reduces resource consumption by dynamically adjusting the number of folds, while still providing reliable performance estimates.

The e-fold cross-validation differs from the traditional k-fold cross-validation by adjusting the number of folds during the validation process. The adjustment is based on four key hyperparameters: e_{max}, e_{count}, $diff_{max}$ and e_{check}. I assume that by tuning these hyperparameters the method can be changed to be either more risk-averse or safer. For example, a more risk-averse approach could use fewer folds and thus stop the method earlier, which could move the model performance further away from the model performance of 10-fold cross-validation. Conversely, a safer approach would use more folds, which may result in model performance closer to the model performance of 10-fold cross-validation. In the following, the method is explained in detail and describes how the above mentioned hyperparameters could affect the results in the context of the e-fold cross-validation method.
The e-fold cross-validation method starts by splitting the entire dataset D into e_{max}, equally sized folds. The hyperparameter e_{max} is comparable to the k value in k-fold cross-validation and specifies the maximum number of folds that can be used. It is important to note that the value of e_{max} cannot be greater than the number of data points in the dataset. The smallest value for e_{max} is $e_{max} = 4$ because e-fold cross-validation can only stop from the third fold at the earliest. However, the value for e_{max} should not be set too small, as a value for e_{max} that is too small would limit the possibility of saving a significant number of folds, which would reduce the advantages of e-fold cross-validation. Based on the literature, which suggests 10 as the optimal value for k-fold cross-validation, I also recommend $e_{max} = 10$ for e-fold cross-validation.
After the dataset D has been divided into e_{max} folds, D consists of $\{f_1, f_2, \ldots, f_{e_{max}}\}$, where each f_i is an equal size part of the dataset. In each iteration e where $e \in \{1, 2, \ldots, e_{max}\}$, the model is trained on 90% of the data $(D - f_e)$ and validated and evaluated on the remaining 10% (f_e). As a result of the evaluation, the performance S_e is stored in a list S, which contains all scores from S_1 to S_e, in other words, S consists of $\{S_1, S_2, \ldots, S_e\}$. M_e is the average of all scores contained in S. M_e shows the aggregated performance of the model at time e.
When $e > 1$, the standard deviation σ_e for samples is calculated. I decided to use the standard deviation because it is a clear and quantifiable measure of the variability of the data. It is also the most widely used and most useful measure of variability [19].
When $e > e_{check}$, e-fold cross-validation checks if the standard deviation σ_e measured in the current iteration is smaller than the standard deviation of the previous iteration, σ_{e-1}.

e_{check} defines the earliest fold from which the standard deviations of the previous folds are compared. I assume that increasing the value of e_{check} will make the e-fold cross-validation safer. If, on the other hand, I choose a low value for e_{check}, I expect the method to be more risk-averse. The minimum value for e_{check} is 2 as e-fold cross-validation can only terminate from the 3rd fold at the earliest, depending on the hyperparameter combination. The maximum value for e_{check} is e_{max}–e_{count}, this means that e_{check} should be set in a way that at least e_{count} more folds are possible after the start of the comparison.
If $e > e_{check}$ and $\sigma_e <= \sigma_{e-1}$, the stability counter variable *count* is increased by 1. If $\sigma_e > \sigma_{e-1}$, e-fold cross-validation checks whether the absolute difference between the current standard deviation σ_e and the previous standard deviation σ_{e-1} is greater than the threshold value defined by the hyperparameter $diff_{max}$.
If the deviation is less than or equal to $diff_{max}$, the stability counter is increased, indicating that there has been no significant change in the model performance. Otherwise, the counter is set to 0. $diff_{max}$ indicates the percentage value at which a deviation from the previous standard deviation is considered constant. I expect that a smaller value of $diff_{max}$ leads to a safer e-fold cross-validation method. Conversely, if the $diff_{max}$ value is increased I expect a more risk-averse e-fold cross-validation method. The value for $diff_{max}$ must be greater than 0 and must not exceed 100. I assume that a value of 5% provides a good reference point, as many statistical methods use a value of 5%. For example, a significance level of 5% is often chosen in statistics [18].
If the stability counter *count* is equal to the parameter e_{count} the cross-validation is stopped, and the e-fold cross-validation method terminates. This implies there are no more significant deviations in the model and it is performing stable. The stability counter e_{count} indicates how often in a row the standard deviation either has to decrease or remain approximately the same to indicate stable model performance. I assume that a higher value for the counter e_{count} would make the e-fold cross-validation method safer. Conversely, I assume that if I choose a smaller value for e_{count}, the e-fold cross-validation method will be more risk-averse. The value of e_{count} should be at minimum 1 because at least one iteration without significant change of the standard deviation is necessary to determine the stability. The value of e_{count} should not be greater than $e_{max} - e_{check}$, as the method only runs until the maximum number of folds or the stability counter is reached.
The e-fold cross-validation method terminates when either the maximum number of folds (e_{max}) is reached or, as just described, the stability counter *count* has reached the value e_{count}, which indicates that the model performance can be considered stable. The average of the previous performances, M_e, at this point, represents the final model evaluation score.
I expect that this dynamic approach to adjust the number of folds (e) considering the hyperparameters e_{max}, e_{count}, $diff_{max}$ and e_{check} will allow a reduction in resource consumption without compromising the reliability of the model's performance evaluation.

In algorithm 1, the e-fold cross-validation process is shown in pseudo-code. The source code for the implementation of the e-fold cross-validation can be found on the attached data medium and can be viewed on GitHub [20].

Algorithm 1: E-fold cross-validation

Data: Dataset D divided into 10 folds $\{f_1, f_2, \ldots, f_{10}\}$
Result: Average model performance M_e

Initialize list $S = []$;
$e_{max} = 10$;
$e_{count} = 2$;
$diff_{max} = 0.05$;
$e_{check} = 2$;
$count$;
for $e = 1$ **to** e_{max} **do**
    Train model on $D - f_e$;
    Validate model on f_e;
    S_e = performance of the model on f_e;
    $S.append(S_e)$;
    $M_e = \frac{1}{e} \sum_{i=1}^{e} S_i$;
    if $e > 1$ **then**
        $\sigma_e = \sqrt{\frac{1}{e-1} \sum_{i=1}^{e} (S_i - M_e)^2}$;
        if $e > e_{check}$ **then**
            if $\sigma_e < \sigma_{e-1}$ **then**
                $count = count + 1$;
            end
            else if $|\sigma_e - \sigma_{e-1}| > diff_{max} \times \sigma_{e-1}$ **then**
                $count = 0$;
            end
            else
                $count = count + 1$;
            end
            if $count == e_{count}$ **then**
                Terminate cross-validation;
            end
        end
    end
end

Unlike other dynamic methods, discussed in detail in Section 2, which focus on the early elimination of poorly performing models based on their average scores, e-fold cross-validation is based on the statistical stability of the model's performance across the folds.
Figure 1 shows how e-fold cross-validation works and how it differs from traditional k-fold cross-validation. On the left side of the diagram, the typical progression of the k-fold cross-validation with k = 10 is shown. For each fold, the performance of the model is evaluated. The 10-fold cross-validation requires 10 folds for an average performance of 0.945.
The right side of Figure 1 shows the e-fold cross-validation process. In addition to the evaluated performance, the e-fold cross-validation also calculates the standard deviation of all previously evaluated performances for each fold. The standard deviation decreases from fold 2 to fold 3 and from fold 3 to fold 4, which means that the stopping criterion is met and e-fold cross-validation stops early. The model performance of 0.944 does not deviate significantly from the model performance of the 10-fold cross-validation and requires 6 folds less than the 10-fold cross-validation.

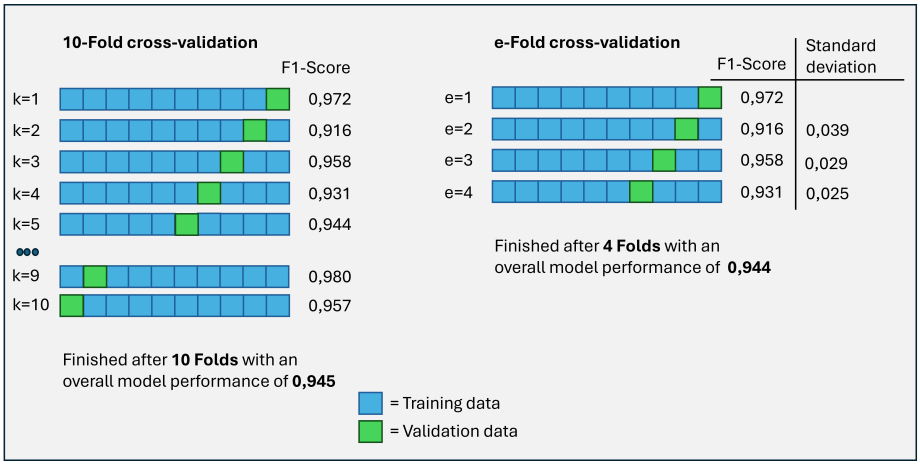

Figure 1: Difference between k-fold cross-validation and e-fold cross-validation

4 Methodology

This section describes the methodological approach used to achieve the defined goal of this thesis. The methodology is divided into three sections, which each address one of the key research questions.
First, the experimental setup and the evaluation process for the comparison between e-fold and 10-fold cross-validation is described in 4.1, which generally forms the basis for all further research questions.
Second, the experimental setup and evaluation process for hyperparameter optimization with e-fold cross-validation in comparison to the hyperparameter optimization with 10-fold cross-validation, is then explained in 4.2.
Finally, the setup and evaluation process for optimizing the hyperparameters of the e-fold cross-validation method is described in 4.3.

4.1 Methodical approach for the comparison between e-fold and 10-fold cross-validation

4.1.1 Experimental setup e-fold cross-validation

I evaluated e-fold cross-validation on 15 datasets (table 1) and 10 algorithms to test an ensure the robustness and generalizability of the e-fold cross-validation method. The following subsections provide a brief general overview of the datasets, algorithms, performance metrics and hyperparameter combinations used for e-fold cross-validation.

Datasets: The 15 datasets comprised of 10 datasets for classification and 5 datasets for regression. The classification tasks included both binary and multi-class scenarios. The datasets varied in size, number of instances and number of classes (table 1), with a focus on small to medium-sized datasets for current and future analyses. Instances of classification datasets were shuffled with scikitlearn's StratifiedKFold function [28] and then randomly split into $e_{max} = 10$ folds. This function also ensured that the class distribution in each fold is approximately the same.
In 3 of the 15 datasets, preprocessing was necessary to ensure that the data was suitable for modeling. In the Diabetes prediction dataset, I converted the textual variable gender to numeric values to make it usable for machine learning analysis. I removed the textual smokinghistory variable, to reduce complexity. In the Air Quality dataset, I removed the RecordID column as it only provided a unique identifier and did not provide any information for modeling. For the same reason, I have removed the StudentID column in the Student performance dataset. I did not perform any additional normalization or scaling of the data, as the focus of the study is on evaluating the method and not on optimizing the models.

Table 1: Supplementary information for all datasets

Classification Datasets				
Dataset Name	Source	Instances	Features	Classes
Mushroom dataset[45]	kaggle	54035	22	2
Diabetes prediction dataset[24]	kaggle	100000	8	2
Heart Failure Prediction - Clinical Records[41]	kaggle	5000	12	2
Breast cancer wisconsin dataset[47]	scikit learn	569	30	2
Phishing Websites[22]	OpenML	11055	31	2
Iris plants dataset[7]	scikit learn	150	4	3
Optical recognition of handwritten digits dataset[33]	scikit learn	1791	16	10
Wine recognition dataset[1]	scikit learn	178	13	3
Student performance[15]	kaggle	2392	13	5
Air Quality and Health Impact dataset[14]	kaggle	5811	13	5
Regression Datasets				
fetch california housing[31]	scikit learn	20640	8	-
Diabetes dataset[32]	scikit learn	442	10	-
elevators[39]	OpenML	16599	19	-
cpu small[40]	OpenML	8192	13	-
ERA[12]	OpenML	1000	4	-

Algorithms: For classification, I selected the following algorithms: AdaBoost, Decision Tree Classifier, Gaussian Naive Bayes, K-Nearest Neighbors and Logistic Regression. For regression, I used the following algorithms: Decision Tree Regressor, K-Nearest Neighbors Regressor, Lasso Regression, Linear Regression and Ridge Regression.
All algorithms were taken from the standard library of scikit-learn [28]. The chosen algorithms represent a diverse set of different modeling approaches. This enabled a comprehensive evaluation of the performance, assessed the robustness and generalizability of the method, and took into account different theoretical foundations on which the algorithms were based. The decision to use these algorithms was made because they cover various modeling approaches and are common in machine learning research.
I decided to use the default parameters for all algorithms and not to use hyperparameter optimization. In the first step, the e-fold cross-validation method is expected to work well regardless of the model's performance.

Performance Metrics: I measured the performance of the binary classification models with the F1-score. For multi-class classification tasks, I used the weighted F1-score. I chose the F1-score because it provides a balanced view of the model's performance [8]. For regression tasks, I used Mean Absolute Error, because it is a robust and easily interpretable measure of model error [46].

Selection of the hyperparameters for e-fold cross-validation: In the following, the choice of the hyperparameters used for the e-fold cross-validation is described. I assume that the choice of the following parameters is a good balance between saving folds and getting as close as possible to the performance of

the 10-fold cross-validation. For e_{max} I chose, as recommended at the beginning, $e_{max} = 10$, as this value is most often considered optimal in the literature. For e_{count}, I chose $e_{count} = 2$ because I assumed that two consecutive iterations with decreasing or stable standard deviation were generally a strong indication that the stability of the model is not random but actually consistent. For $diff_{max}$ I chose $diff_{max} = 0.05$, as described in Section 3. For the parameter e_{check} I chose $e_{check} = 2$, which means that the e-fold cross-validation can stop from the 4th fold at the earliest. I assume that e-fold cross-validation can deliver statistically significant results after 4 folds at the earliest. Furthermore, the goal was to save as many folds as possible, and therefore selecting the smallest possible value for e_{check}, in the given hyperparameter combination, was considered a good choice.

4.1.2 Evaluation criteria e-fold cross-validation

The performance of 10-fold cross-validation was considered as ground truth. This means that e-fold cross-validation was expected to come as close as possible to this ground truth. It is important to note that 100 runs were performed for each algorithm on each dataset, with a different data distribution within the folds. As a result, with the 15 datasets and 10 algorithms, there were a total of 75 algorithm-dataset combinations, with each combination having 100 iterations with a specific algorithm on a dataset. In total, for all 75 combinations, this resulted in 7500 iterations. For some parts of the evaluations, the average value of the 100 runs was used. In detail, the performance of e-fold cross-validation was evaluated as follows.

Percentage performance difference: First, I checked how much the performance of e-fold cross-validation deviates from the performance of 10-fold cross-validation. In this way, I could measure how accurate the score of the e-fold cross-validation method is compared to the ground truth, despite the early stopping. For the evaluation, I calculated the absolute average difference in percent between the performance metric of the 10-fold cross-validation $M_{e_{max}}$ and the performance metric of the e-fold cross-validation M_e. The optimal result that can be achieved is a difference of 0%, meaning no discrepancy.
However, it is important to note that there may be situations in which early stopping does not always offer a significant advantage. For example, if the differences between the results of the e-fold cross-validation and the ground truth are statistically significant, early termination could lead to an under- or overestimation of the model's performance. In this case, additional iterations would be required to obtain a performance estimate that is closer to the actual k-fold cross-validation result. Furthermore, if the computational resources required to perform all folds are minimal, the risk of less accurate performance estimation could be avoided by early termination.

Saved folds and resources: Second, I checked how many folds were saved by stopping the cross-validation earlier. This corresponds to $e_{max} - e$ saved folds which meant that e-fold-cross-validation needed $e_{max} - e$ fewer resources such as evaluation time and energy consumption. In this setup, e-fold cross-validation can terminate earliest at $e = 4$, which is the optimum.

Statistical tests: Finally, I showed the statistical significance of the results by evaluating whether the early stopped score M_e falls within the confidence interval [34]. The confidence interval was calculated from all e_{max} scores in S. For confidence of 95%, I used the standard deviation σ_k of the scores and the t-value for the 95% confidence level. The score M_e was considered good if it fell within the upper and lower bounds of the confidence interval.

The following Figures (2 - 4) provide a good overview of the evaluation of the e-fold cross-validation based on real runs of this experiment. Figure 2 illustrates a successful process of e-fold cross-validation. The blue line represents the individual F1-scores of each iteration, while the green line indicates the average F1-score for each iteration. The yellow dashed line shows the mean score, the upper and lower boundaries of the 95% confidence interval, calculated from the results of all 10 single scores. This example shows that the individual results converge after each fold, leading to a decrease in the standard deviation for two consecutive folds. Consequently, the stopping criterion is met, and the red dashed line marks the early stopping after 4 folds.

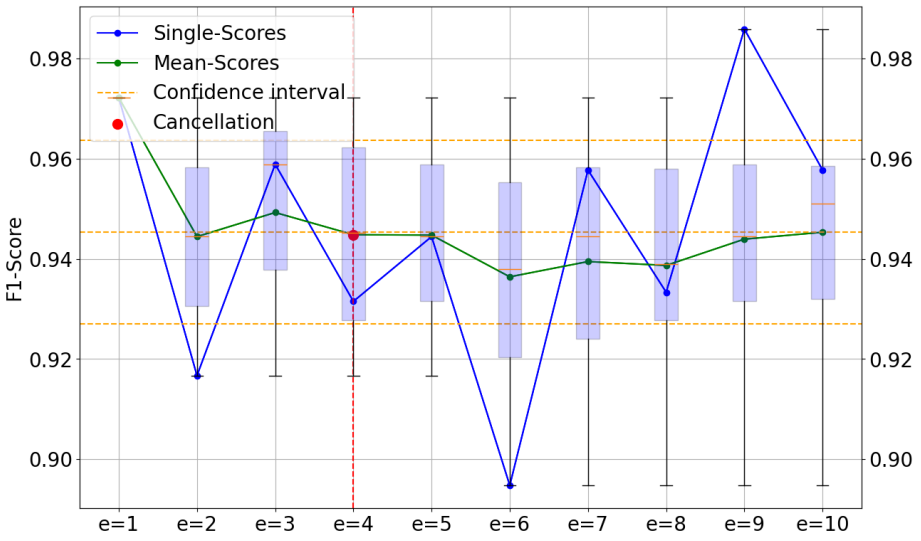

Figure 2: Evaluation of e-fold cross-validation based on real runs of the experiment

In contrast, Figure 3 shows an example in which the e-fold cross-validation does not stop early. Due to the constantly varying individual scores, the standard deviation does not decrease continuously, so the stopping criterion is not met.

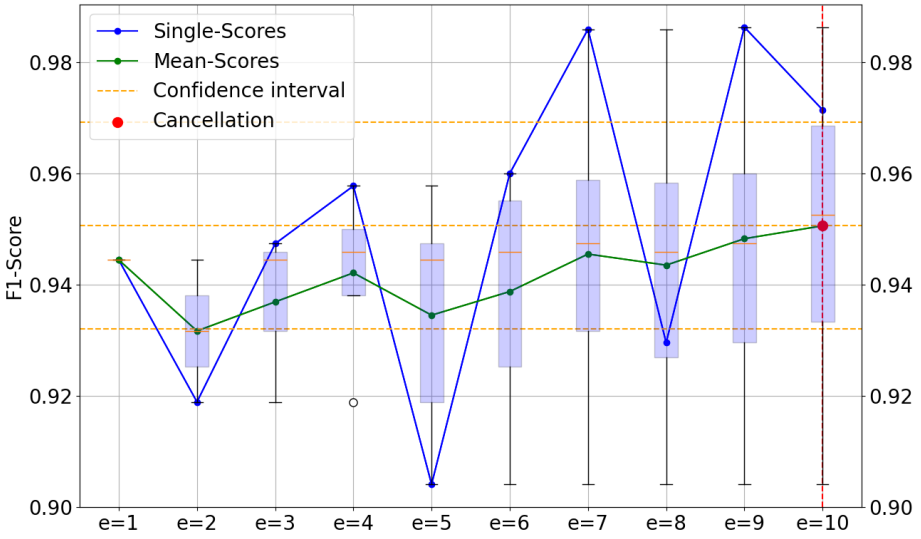

Figure 3: Evaluation of e-fold cross-validation based on real runs of the experiment

Figure 4 shows that the stopping criterion for e-fold cross-validation is met, but the F1-score is outside the confidence interval. Therefore, the F1-score deviates significantly from the measured $k = 10$ score. Due to the consistent F1-score in the first 4 folds, the stopping criterion is met early, but the following folds perform with a higher F1-score.

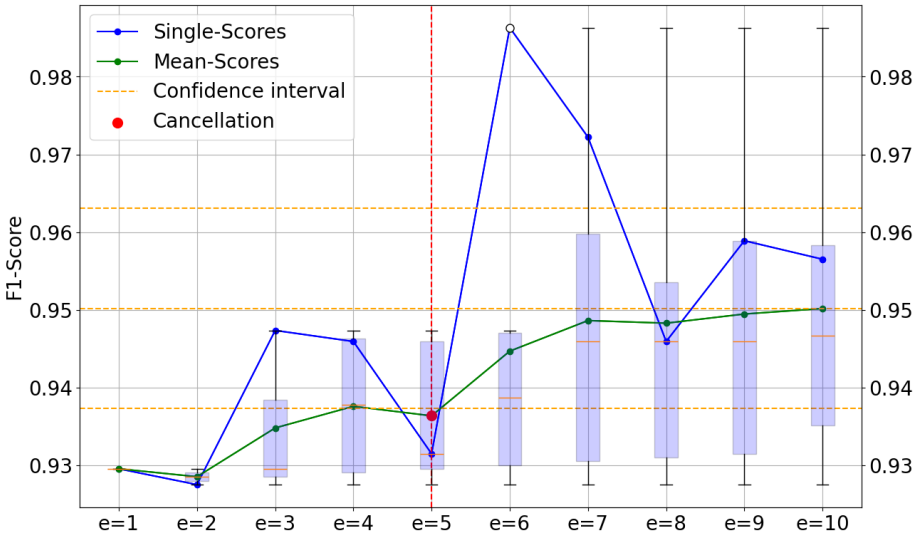

Figure 4: Evaluation of e-fold cross-validation based on real runs of the experiment

4.2 Methodical approach for the hyperparameter optimization using e-fold cross-validation compared to using 10-fold cross-validation

4.2.1 Experimental setup hyperparameter optimization using e-fold cross-validation

For the hyperparameter optimization, I used grid search because grid search systematically explores the entire hyperparameter configuration space [4]. Grid search is particularly beneficial as it allows a complete exploration of all possible combinations of the specified hyperparameters. This ensures that no potentially optimal configuration is missed and that the best possible performance for the model is found, based on the given parameters.

Datasets: For the hyperparameter optimization with e-fold cross-validations, I used the datasets from 4.1.1, except the Student performance and Air Quality datasets. I did not use these 2 datasets for technical performance reasons and the associated time expenditure. In total, I performed the hyperparameter optimization on 13 datasets including 8 classification datasets and 5 regression datasets. The dataset was divided into a train/test split. 80% train data and 20% test data. The 20% test data are used to evaluate the performance of the best founded hyperparameter combinations on new unseen data, which allows a more realistic assessment of the model.

Algorithms: I also used the algorithms as described in section 4.1.1, except Gaussian Naive Bayes and Lasso Regression. Gaussian Naive Bayes has no hyperparameters that can be optimized, so I did not perform hyperparameter optimization for it. I also decided not to use Lasso Regression for technical performance reasons and the associated time expenditure. I have performed hyperparameter optimization for all other algorithms. For the choice of configuration space, I orientated on a GitHub repository for Auto-Sklearn [6]. These configurations are based on best practices and are often used to provide a solid foundation for optimizing algorithms. The specific hyperparameters that I tested for the various algorithms are listed in the following:

K-Nearest Neighbors and Regressor hyperparameters:

Hyperparameter	**Tested Values**
n_neighbors	1, 2, 3, 4, 5, 7, 9, 11, 14, 18, 23, 30, 38, 48, 62, 78, 100
weights	uniform, distance
p	1, 2

Table 2: Hyperparameter search space for K-Nearest Neighbors

Decision Tree Classifier hyperparameters:

Hyperparameter	Tested Values
Criterion	gini, entropy
Max Depth Factor	1, 10, 20, 30
Min Samples Split	2, 6, 10, 14, 18, 22, 26, 30

Table 3: Hyperparameter search space for Decision Tree Classifier

Logistic Regression hyperparameters:

Hyperparameter	Tested Values
C	10 logarithmically spaced values between 10^{-4} and 10^{4}
solver	lbfgs, saga
max_iter	100, 250, 500, 1000

Table 4: Hyperparameter search space for Logistic Regression

AdaBoost hyperparameters:

Hyperparameter	Tested Values
n_estimators	50, 100, 200,300,500
learning_rate	0.01, 0.03, 0.1, 0.3, 1.0, 2.0

Table 5: Hyperparameter search space for AdaBoost

Decision Tree Regressor hyperparameters:

Hyperparameter	Tested Values
Max Depth Factor	1,3,4,5,10
Min Samples Split	2, 6, 10, 14, 18, 20

Table 6: Hyperparameter search space for Decision Tree Regressor

Ridge Regression hyperparameters:

Hyperparameter	Tested Values
alpha	100 logarithmically spaced values between 10^{-8} and 10^{8}

Table 7: Hyperparameter search space for Ridge Regression

Linear Regression hyperparameters:

Hyperparameter	Tested Values
fit_intercept	True, False
copy_X	True, False
positive	True, False

Table 8: Hyperparameter search space for Linear Regression

Performance Metrics: To evaluate the model performance, I used the same performance metrics as in section 4.1.1.

Selection of the hyperparameters for e-fold cross-validation: For the hyperparameter optimization with e-fold cross-validation, I used the same hyperparameter for the e-fold cross-validation method described in 4.1.1.

4.2.2 Evaluation criteria for hyperparameter optimization using e-fold cross-validation

I considered the performance of hyperparameter optimization using 10-fold cross-validation as a ground truth. That means I expect the hyperparameter optimization with e-fold cross-validation to come as close as possible to this ground truth. It is important to note that I evaluated each hyperparameter optimization per algorithm per dataset 25 times with a different data distribution to show the robustness of the results. This means that 52 algorithm-dataset combinations were evaluated, each executed 25 times with a different data distribution. Based on the hyperparameter combinations of all algorithms, the e-fold cross-validation is applied 74150 times as part of the hyperparameter optimization. Specifically, I evaluated the performance of hyperparameter optimization with e-fold cross-validation as follows:

Saved folds and resources: First, I checked how many folds were saved by using e-fold cross-validation instead of 10-fold cross-validation to optimize the hyperparameters. I, therefore, checked for each hyperparameter combination how many folds the e-fold cross-validation required to evaluate the model performance of the chosen combination. After evaluating all combinations, I calculated the average number of folds needed for all combinations. The optimum value that can be achieved is 4 fold, which means a saving of 60% of resources.

The same best found hyperparameter combination: Second, I checked if the hyperparameter optimization with e-fold cross-validation suggests the same best hyperparameter combination as the hyperparameter optimization with 10-fold cross-validation. The optimum that can be achieved is that the hyperparameter optimization with e-fold cross-validation suggests the exact same best hyperparameters as the hyperparameter optimization with 10-fold cross-validation for all 25 runs for each algorithm-dataset combination.

Percentage deviation of the best hyperparameter combinations: Finally, I selected the best found hyperparameter combination of all runs of all algorithm dataset combinations, once based on e-fold cross-validation and once based on 10-fold cross-validation. I then evaluated the model performance with these combinations on the previously unseen 20% test data from each dataset to assess how well the model performed on this data. From this, I calculated the performance metric based on the best found hyperparameters using e-fold cross-validation and once based on the best found hyperparameters using 10-fold cross-validation. Afterwards, I calculated the absolute percentage deviation from these performance metrics. The equal best found hyperparameters are not taken into account.

4.3 Methodical approach to investigate the impact of hyperparameter adjustment for e-fold cross-validation

4.3.1 Experimental setup hyperparameter adjustment for e-fold cross-validation

Datasets: I used the same 15 datasets from table 1 and the same preprocessing described in 4.1.1 for the hyperparameter optimization with e-fold cross-validation.

Algorithms: I used the same algorithms and setups for the algorithms described in 4.1.1.

Performance Metrics: I used the same performance metrics described in 4.1.1.

4.3.2 Evaluation criteria for the impact of hyperparameters adjusting for e-fold cross-validation

This section describes how the individual hyperparameters of e-fold cross-validation could affect the results and the approach used to analyze them. The evaluation was based on the same methodology as described in 4.1.2. Specifically, this meant that I tested how different values for the hyperparameters e_{max}, e_{count}, $diff_{max}$ and e_{check} of the e-fold cross-validation method influenced the number of saved folds, the percentage deviation of the performance metric compared to 10-fold cross-validation and the number of iterations in the confidence interval. I chose the values for the hyperparameters as follows:

e_{max}: In this bachelor thesis, I have decided not to analyze the influence of the parameter e_{max} any further. There are two main reasons for this: First, $e_{max} = 10$ is a frequently recommended value in the literature, which is considered a good baseline for k-fold cross-validation methods. Second, a detailed investigation to determine the optimal value for e_{max} or k in k-fold cross-validation is out of the scope of this thesis. Therefore, I set $e_{max} = 10$ for all further analyses.

$diff_{max}$: I have also decided not to perform a detailed analysis for the $diff_{max}$ parameter. The value of 5% for $diff_{max}$ is a frequently used value for statistics, which was adopted in this work. Therefore, $diff_{max} = 0.05$ was left unchanged for the evaluation of the other parameters.

e_{count}: Depending on the chosen value for e_{count}, I expected different trade-offs between the number of saved folds, the deviation of the model performance compared to the 10-fold cross-validation and the statistical significance of this result by the number of iterations within the confidence interval. I tested several combinations for the e_{count} parameter to evaluate its influence on the e-fold

cross-validation. Therefore $e_{max} = 10$, $diff_{max} = 0.05$ and $e_{check} = 3$ remained constant for all combinations. The tested values for e_{count} were:
$e_{count} = 1$, an expected risk-averse approach where e-fold cross-validation is stopped after a single iteration if the standard deviation has decreased or remained the same. This setting could lead to more saved folds, but the results could be more influenced by chance and thus be further away from the 10-fold cross-validation performance and have fewer iterations within the confidence interval.
$e_{count} = 2$, an expected balanced approach where two consecutive iterations with decreased or unchanged standard deviation were required to complete the method. This should ensure a certain stability of the results while still allowing some folds to be saved.
$e_{count} = 3$, an expected more conservative setting where three consecutive iterations with decreased or unchanged standard deviation are required before the e-fold cross-validation stops. I expected that this would reduce the risk of random results and could therefore improve the percentage deviation from the 10-fold and the number of iterations in the confidence interval. However, it could lead to fewer saved folds.

e_{check}: By adjusting the parameter of e_{check}, I expected a similar behavior as with e_{count}, a lower value is riskier and a higher value is safer. I tested several combinations for the e_{check} parameter to evaluate its influence on the e-fold cross-validation. Therefore $e_{max} = 10$, $diff_{max} = 0.05$ and $e_{count} = 2$ remained constant for all combinations. The tested values for e_{check} were:
$e_{check} = 2$, this adjustment enables an early check of the standard deviation from the second iteration. This could lead to a maximum reduction in the number of required folds, but there is a risk that the method will terminate too early and the model performance will not be evaluated with sufficient stability.
$e_{check} = 3$, with this setting, the standard deviation is checked from the third iteration onwards. I expected this to be a moderate approach that still offers the possibility to save folds, but at the same time ensures that the e-fold cross-validation is not terminated too early to guarantee stable results.
$e_{check} = 5$, is an expected more conservative setting that postpones the standard deviation check until the fifth iteration. This minimizes the risk of stopping the cross-validation too early and could enable a more accurate model evaluation. However, I expected fewer saved folds here compared to the lower values for e_{check}.

5 Results

5.1 E-fold cross-validation compared to 10-fold cross-validation

In 96.4% of all algorithm-dataset combinations, the results were within the 95% confidence interval. The percentage of 96.4% was determined by analyzing the results of 100 iterations for each of the 75 combinations, with 7230 out of 7500 iterations falling within the confidence interval. Figure 5 illustrates the results of this analysis and shows the evaluation by dataset on the left-hand side and by algorithm on the right-hand side. The x-axis of both diagrams shows the percentage of iterations that fall within the 95% confidence interval, with a value range of 0 to 100. The y-axis of the left graph lists the 15 datasets used, while the y-axis of the right graph shows the 10 algorithms used. The red dashed line represents the 96.4% average score.
For each dataset, the average percentage of iterations that fall within the confidence interval across all algorithms tested on this dataset is displayed in the left-hand graph. It can be observed that the two datasets Iris plants and Wine recognition from the area of multi-class classification and the cpu small dataset from the area of regression deviate more significantly from the average value than other datasets. It is also noticeable that, except for the Phishing Websites dataset, all other classification datasets have a similar result.
The graph on the right shows the average proportion of iterations for each algorithm that falls within the confidence interval across all datasets suitable for this algorithm. It is noticeable here that most algorithms have similar iterations within the confidence interval.

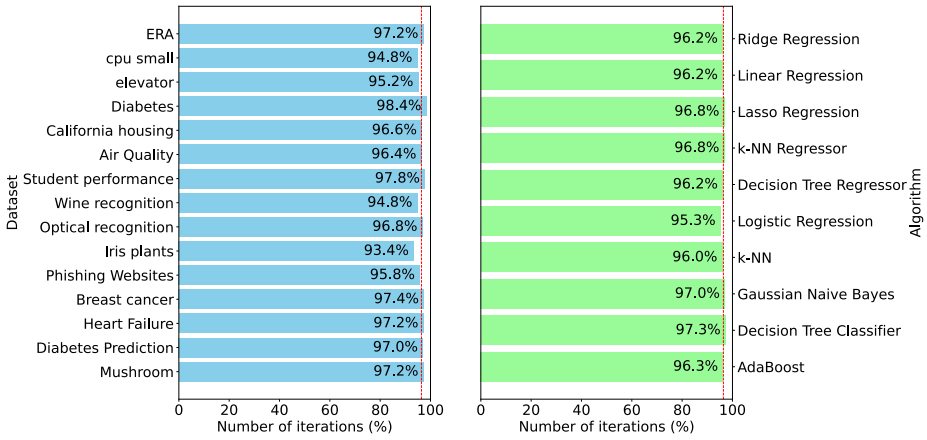

Figure 5: Average percentage number of iteration within the confidence interval for all datasets and all algorithms

Figure 6 complements this analysis by showing the distribution of the number of iterations that fall within the 95% confidence interval across all 75 different algorithm-dataset combinations. The x-axis shows the percentage number of iterations that fall within the confidence interval. The y-axis represents the percentage frequency of these values, i.e. how often a certain number of iterations occurred within the confidence interval across all 75 combinations. A distinct accumulation is observed between 94% and 99% of all iterations, with a particularly notable peak at 98% of all iterations, which occurs in 28% of combinations, making it the most common result.
This results illustrate the accuracy and reliability of the e-fold cross-validation method and shows that the results are statistically significant.

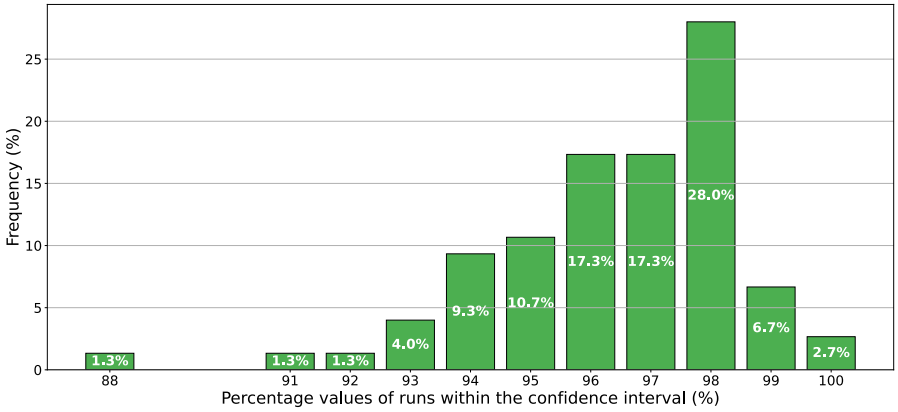

Figure 6: Distribution of the number of iterations within the 95% confidence interval for 75 algorithm-dataset combinations

On average, e-fold cross-validation was completed after 5.7 folds, as shown in Figure 7. Compared to the conventional 10-fold cross-validation, an average saving of 4.3 folds per algorithm and dataset was achieved. This corresponds to a reduction in the required resources, such as evaluation time and energy consumption, of around 40%. Figure 7 shows a detailed view of the results of the average number of folds after which the e-fold cross-validation was stopped. The x-axis of the heatmap in Figure 7 represents the different datasets used, while the y-axis shows the different algorithms used. The color coding of the heatmap reflects different value ranges, with darker shades indicating a low number of required folds and lighter shades indicating a high number of required folds. The numbers shown in the cells represent the average number of folds after which the e-fold cross-validation stops per algorithm per dataset. These average values are calculated from the average of all 100 runs per the respective algorithm-dataset combination. In addition, the average number of folds across all algorithms per dataset and the average number of folds across all datasets per algorithm are displayed, as well as the overall average across all algorithm-dataset combinations.

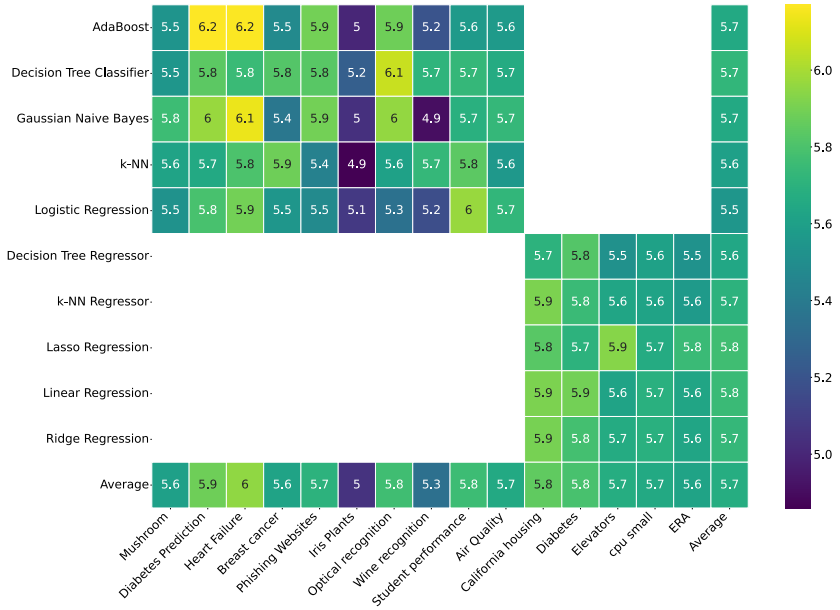

Figure 7: Heatmap of the average number of folds after which e-fold cross-validation was stopped for all datasets and algorithms

The heatmap in Figure 7 also shows that the number of folds required for e-fold cross-validation varies depending on the combination of the dataset and algorithm. For example, datasets Iris Plants and Wine recognition showed a

significantly lower average number of folds required 5.0 and 5.3 respectively compared to the overall average. Interestingly, these are the exact two datasets that scored the worst in the confidence interval score. Datasets Diabetes Prediction and Hearth Failure, on the other hand, show a slightly higher usage of folds compared to the average. The average required folds across all algorithms, on the other hand, show only very slight deviations from the overall average value and are therefore very stable over the entire experiment.

Furthermore, Figure 8 shows that the percentage distribution after how many folds e-fold cross-validation has stopped across all iterations. The x-axis shows the number of folds after which e-fold cross-validation stopped. The y-axis shows the number of all iterations of this experiment. The results show that e-fold cross-validation was terminated after 4 folds in 35.7% of cases and thus the clear majority. In 4.8% of cases, early termination was not possible, resulting in the same number of folds as with 10-fold cross-validation, showing that e-fold cross-validation requires significantly fewer folds than 10-fold cross-validation.
The results demonstrate the dynamics of the e-fold cross-validation method and underline the key advantage that the number of folds is determined dynamically. These results show the potential savings of e-fold cross-validation in terms of computing resources and the associated resource consumption.

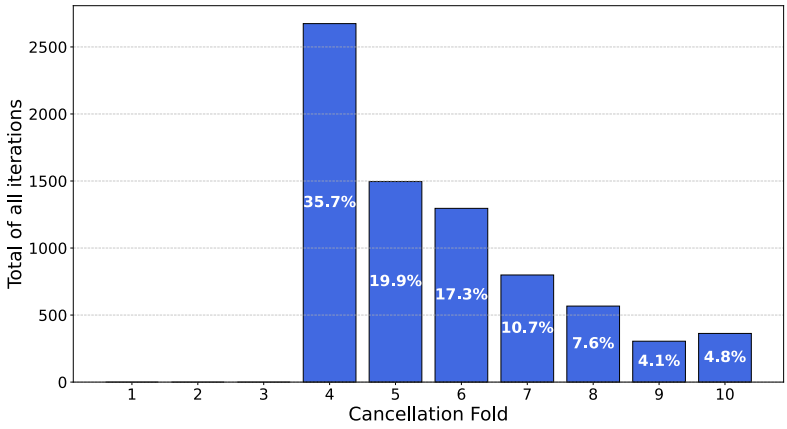

Figure 8: Early stopping using e-fold cross-validation after folds across all 7500 iterations

The results show that despite the early termination, the absolute percentage difference in performance metrics between the e-fold cross-validation and the 10-fold cross-validation remained consistently low. The average performance metric determined by the e-fold cross-validation deviates on average by 0.9% from the performance metric determined by the 10-fold cross-validation across all algorithm-dataset combinations (Figure 9). This value is calculated from the average deviation over all 100 iterations across all algorithm-dataset combinations. Figure 9 shows the average deviation between all 75 algorithm-dataset combinations compared to traditional 10-fold cross-validation in a box plot. The dots indicate the average performance of the respective combination, whereby the dots are color coded depending on the dataset. The y-axis describes the percentage deviation of the model performance from the e-fold cross-validation compared to the 10-fold cross-validation in percent. The red dashed line represents the mean score of 0.9%, and the yellow line the median of 0.7%.

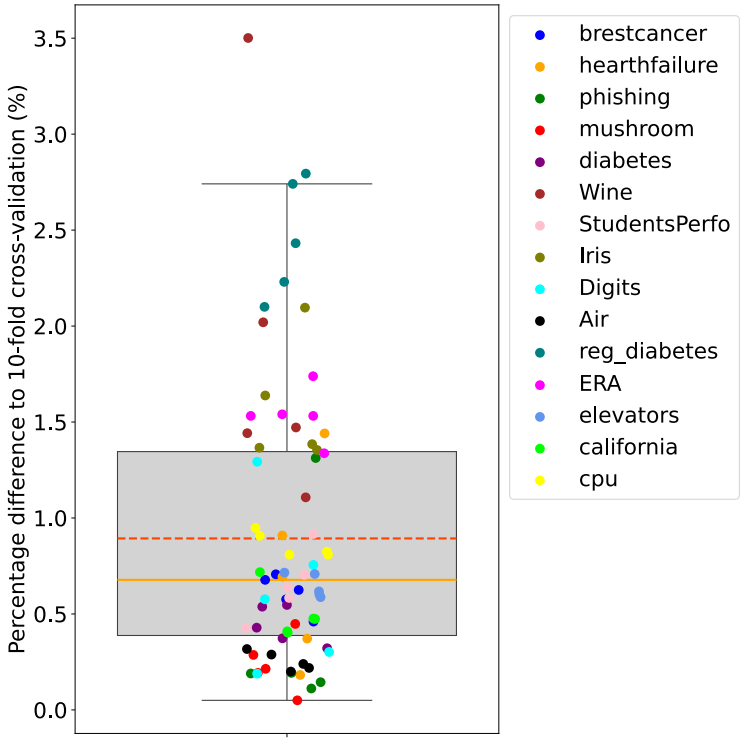

Figure 9: Boxplot of the percentage difference of the performance metric between 10-fold and e-fold cross-validation for all algorithm-dataset combinations

The majority of the data points are below 1%, which again illustrates that the majority of the results are very close to the performance of the 10-fold cross-

validation. There are only 2 outliers with over 3.5% and 2.6%. The boxplot shows that the Wine Recognition, Diabetes (regression), Iris plants and ERA datasets deviate more from the median and the average. Most of their values are outside the box, which indicates that their deviations are outside the normal range of the data distribution. It is interesting to note that these datasets contain fewer instances.
In contrast, the Phishing Websites, Mushroom and Air Quality datasets show a stronger representation in the lower part of the boxplot, within the lower whisker. This means that these datasets have a very low percentage deviation from the 10-fold cross-validation. Interestingly, these datasets have a larger number of instances.

Figure 10 shows the percentage difference per algorithm-dataset combination divided into binary, multi-class and regression datasets. The x-axis shows the respective datasets while the y-axis shows the percentage difference of the performance metric compared to the 10-fold cross-validation.
On average, the binary classification datasets have a lower percentage difference than the multiclass and regression datasets. For the binary classification datasets, this difference averaged less than 1%, with outliers of up to 5%. Similar results were obtained for the multi-class classification and regression datasets, with an average difference of less than 2% with outliers of up to 16% for multi-class classification and up to 12% for regression (Figure 10).
In addition, the diagram of the percentage deviation of the binary classification datasets shows that the algorithms on the respective dataset have approximately the same percentage deviation and the distribution of the boxplot is almost the same, except for the datasets Heart failure and Phishing Websites, where Gaussian Naive Bayes deviates significantly from the other algorithms.
The diagram for the percentage deviation of the multi-class classification datasets shows, similar to the binary class diagram, that the algorithms have a similar percentage deviation, apart from the K-Nearest Neighbors algorithm for the Wine recognition dataset. In addition, the results show again, as previously in Figure 9, that the Iris plants and Wine recognition datasets have a significantly higher percentage deviation than the other datasets.
The diagram of the percentage deviation of the regression datasets again shows that the algorithms on the respective dataset have a similar percentage deviation. Here the same as in Figure 9, the Diabetes dataset has a significantly higher percentage deviation than the other datasets and a wider distribution of data points.

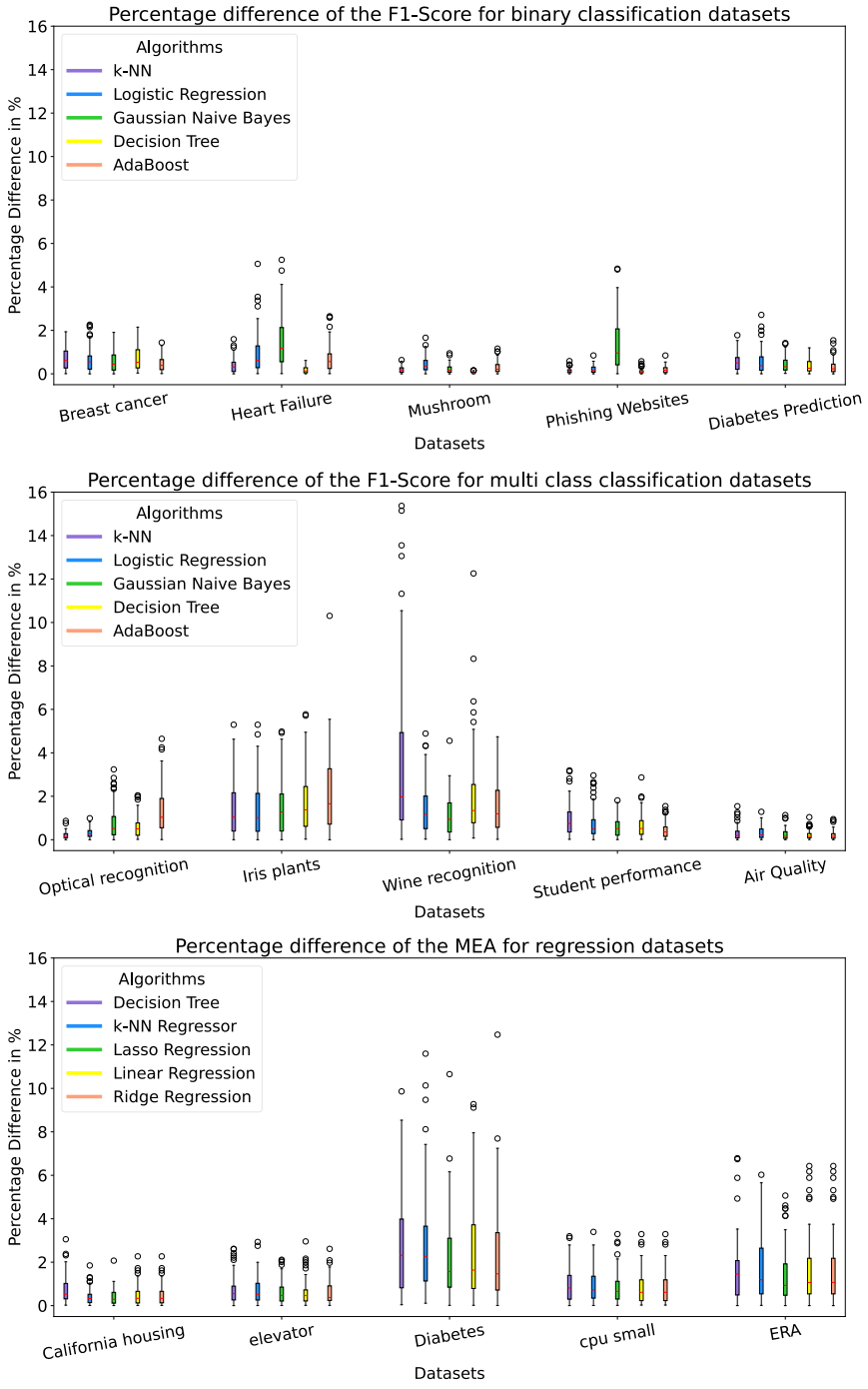

Figure 10: Percentage difference of the performance metric between 10-fold and e-fold cross-validation over 100 runs for all datasets and algorithms

To summarise, these results answer the first key research question. In a direct comparison with 10-fold cross-validation, e-fold cross-validation achieved a deviation of just 0.9% in the performance metric compared with 10-fold cross-validation and saving of 4.3 folds, with 96.4% of the results falling within the confidence interval. The e-fold cross-validation has clearly shown that about 40% of resources can be saved without significantly reducing the accuracy of the performance metric. It is noticeable that the e-fold cross-validation works particularly well with larger datasets, which have more potential to save resources. Smaller datasets, specifically Iris plants and Wine recognition from the multi-class datasets, performed worse than other datasets in terms of both percentage deviation and the number of iterations within the confidence interval. This suggests that e-fold cross-validation is stopped too early on multi-class datasets with few instances, as indicated by the low number of folds, which has a negative impact on other performance criteria. Furthermore, the influence of the datasets on the results was significantly greater than the choice of algorithm. On average the algorithms very often achieved similar results.

5.2 Hyperparameter optimization using e-fold cross-validation compared to using 10-fold cross-validation

For the given hyperparameter search space, the hyperparameter optimization with the e-fold cross-validation found the same best hyperparameter combination in 55.5% as the hyperparameter optimization with the 10-fold cross-validation across all algorithm-dataset combinations. In other words, out of all 1300 possible equal combinations, the hyperparameter optimization with e-fold cross-validation identified the same best combination 722 times. Figure 11 shows a detailed overview of this result. The x-axis of the stacked bar chart shows the percentage number of the correct predicted best hyperparameter configurations, by the hyperparameter optimization with e-fold cross-validation. The y-axis shows the datasets used. An entire bar in the diagram shows how many equal hyperparameter combinations were found for a particular dataset. The different colors within the bar represent the algorithms used and represent how many correct combinations for this specific algorithm were predicted correctly on the particular dataset.

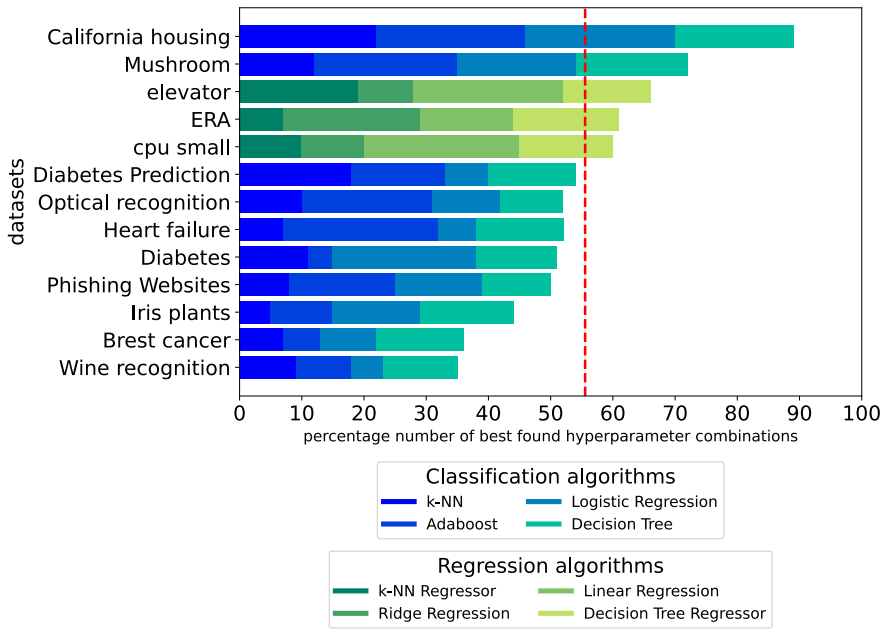

Figure 11: Number of identical best hyperparameter combinations found for hyperparameter optimization using e-fold cross-validation compared to hyperparameter optimization using 10-fold cross-validation

The diagram shows that the hyperparameter optimization with e-fold cross-validation found 89% of the same best hyperparameter combinations on the California housing dataset, which was the highest result achieved. It can be observed that for all algorithms on this dataset, a similar number of correct identified combinations were found. The Mushroom dataset shows similar results, but fewer equal combinations were found in the K-Nearest-Neighbors algorithm. It is particularly noticeable that for the regression datasets and algorithms, the hyperparameter optimization with e-fold cross-validation found more correct combinations compared to the classification datasets and algorithms. The correct hyperparameter combinations were particularly often correctly identified in the Linear Regression and Decision Tree Regressor algorithms for regression tasks and in the AdaBoost and Decision Tree Classifier algorithms in the classification tasks. Only 35 and 36 correct hyperparameter combinations were identified on the Wine recognition and Breast cancer datasets, significantly below the average.
However, the lower number of combinations found does not necessarily indicate that the hyperparameters searched with e-fold cross-validation for these algorithm-dataset combinations deviate significantly from the hyperparameters found with 10-fold cross-validation. Figure 12 shows the hyperparameters found for the K-Nearest-Neighbors algorithm on the Wine recognition dataset in the hyperparameter optimization with e-fold cross-validation in blue and with 10-fold cross-validation in orange.

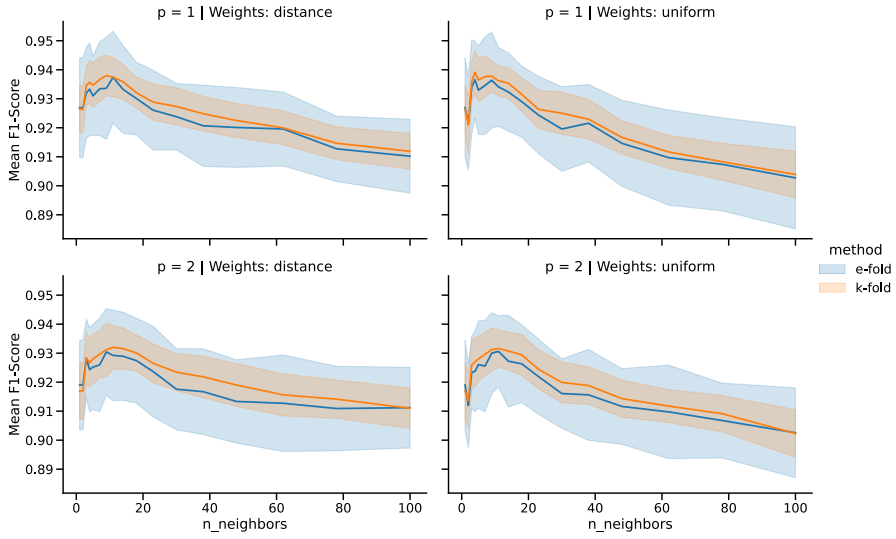

Figure 12: Hyperparameter optimization with e-fold cross-validation and 10-fold cross-validation for the K-Nearest Neighbors algorithm on the Wine recognition dataset

The diagram shows that the combinations found on average by the hyperparameter optimization with e-fold cross-validation are very similar to the combinations found by the hyperparameter optimization with 10-fold cross-validation. The blue and orange areas indicate the standard deviation of the combinations found, with the blue area deviating slightly more. Nevertheless, it can be seen that the hyperparameter space was searched similarly well.

Figure 11 shows that only 7 correct combinations were found for the K-Nearest Neighbors algorithm on the Phishing Websites dataset. Figure 13 describes the hyperparameter optimization with e-fold cross-validation for the K-Nearest Neighbors algorithm on the Phishing websites dataset, similar to Figure 12. Figure 13 shows that the hyperparameter space was searched almost identically by the hyperparameter optimization with e-fold cross-validation compared to the hyperparameter optimization with 10-fold cross-validation. These results show that the hyperparameter optimization with e-fold cross-validation does not always find the same optimal combination as the hyperparameter optimization with 10-fold cross-validation, but that the hyperparameter search space was searched in a very similar way.

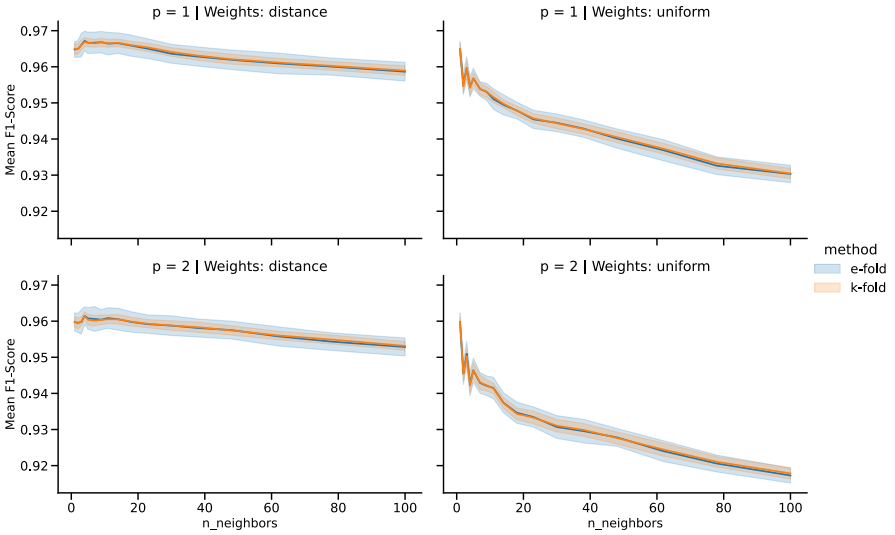

Figure 13: Hyperparameter optimization with e-fold cross-validation and 10-fold cross-validation for the K-Nearest Neighbors algorithm on the Phishing Websites dataset

The remaining 44.5% of the best hyperparameter combinations that were not identically identified by the hyperparameter optimization with e-fold cross-validation were evaluated on the unseen test data. The resulting performance metric was compared with the performance metric based on the best found hyperparameters of the hyperparameter optimization with 10-fold cross-validation, which were also evaluated on the test data, and then the absolute percentage deviation between these two metrics was calculated.
On average across all algorithm-dataset combinations, this percentage deviation was less than 2%.
Figure 14 and 15 describe this percentage deviation in more detail. In Figure 14 the percentage deviation is shown for all classification datasets using a boxplot. The individual points represent the specific algorithms. The x-axis shows the datasets used. The y-axis shows the percentage deviation compared to the performance metric of the best found hyperparameter combination evaluated on the unseen test data. The first diagram shows that the deviation rarely exceeds 2%. In the diagram below, a larger deviation is recognizable, especially on the Iris plants dataset in combination with the AdaBoost algorithm, which has a deviation of over 10%.
Figure 15 shows the percentage differences of the regression datasets. The regression datasets also show a low percentage deviation for all datasets except the Diabetes dataset. The Diabetes dataset in combination with K-Nearest-Neighbors Regressor and Decision Tree Regressor algorithms has high outliers with almost 20% deviation.
The larger deviation could indicate that the e-fold cross-validation in the hyperparameter optimization was stopped too early and therefore a significantly worse hyperparameter combination was considered optimal. On the other hand, the results of the small percentage difference could indicate that the hyperparameter optimization with e-fold cross-validation has found suitable hyperparameters and could also be an indication that in most cases the hyperparameter space was analyzed very similarly to the hyperparameter optimization with 10-fold cross-validation, as also shown earlier in Figure 12 and 13.

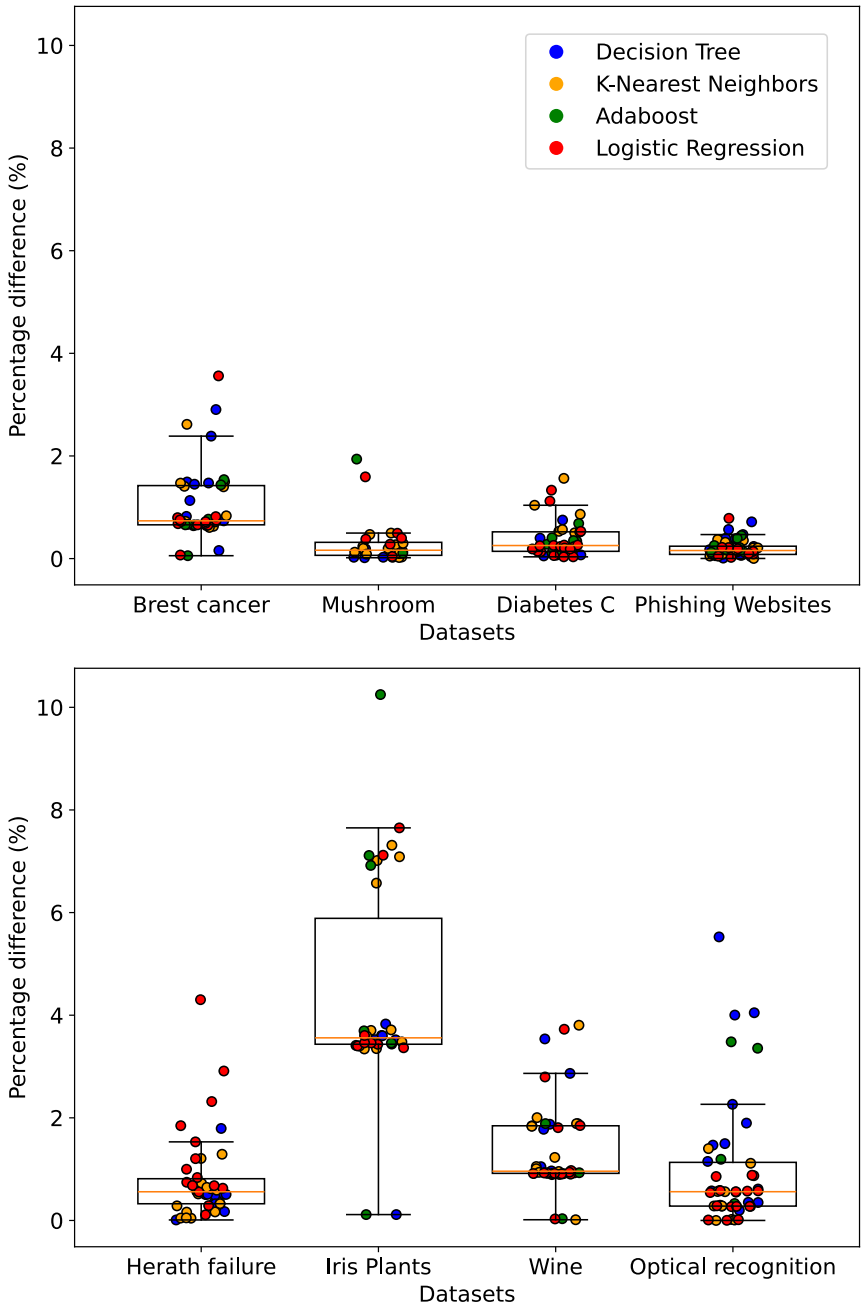

Figure 14: Percentage deviation of the best hyperparameter combinations found with hyperparameter optimization with e-fold cross-validation compared to the best hyperparameter combinations with 10-fold cross-validation on the binary classification test datasets

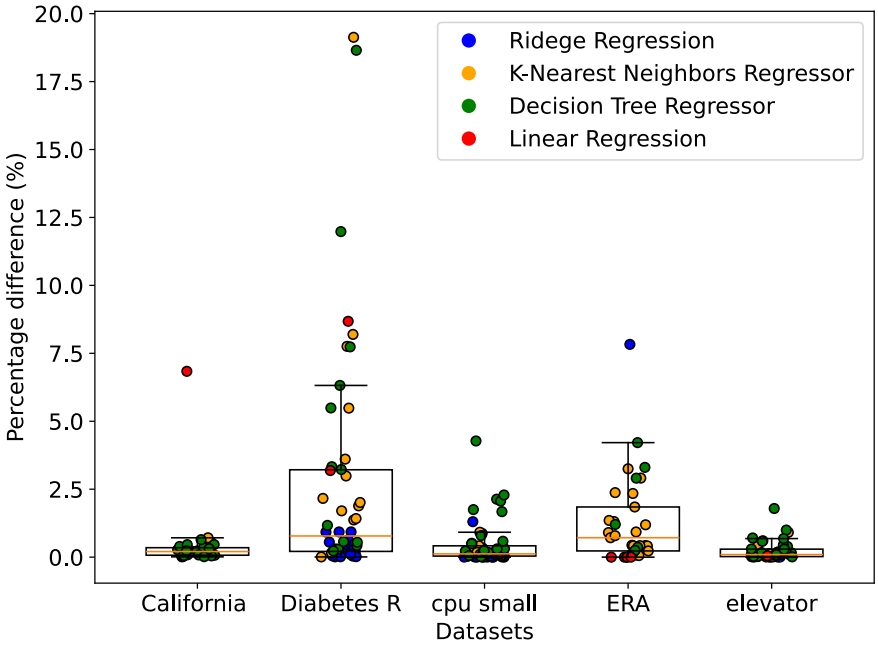

Figure 15: Percentage deviation of the best hyperparameter combinations found with hyperparameter optimization with e-fold cross-validation compared to the best hyperparameter combinations with 10-fold cross-validation on the regression test datasets

On average, the hyperparameter optimization with e-fold cross-validation needed 5.6 folds to evaluate the hyperparameter configuration spaces as shown in Figure 16. Compared to the hyperparameter optimization with 10-fold cross-validation, the hyperparameter optimization with e-fold cross-validation required 4.4 folds less, which is a saving of about 40% in resources such as evaluation time and energy consumption. Figure 16 shows detailed in a heat map, similar to the illustration in 5.1 (Figure 7), how many folds were required for each combination of algorithm and dataset. The color coding and axis assignment correspond to that in section 5.1. Except for the datasets Iris plants with 5.1 needed folds on average, all datasets and algorithms show similar savings. The lower number of required folds could again indicate that e-fold cross-validation was stopped too early, which could explain why the hyperparameters found on the Iris plants dataset had a high percentage deviation on the one hand and on the other hand found few equal best hyperparameter combinations.

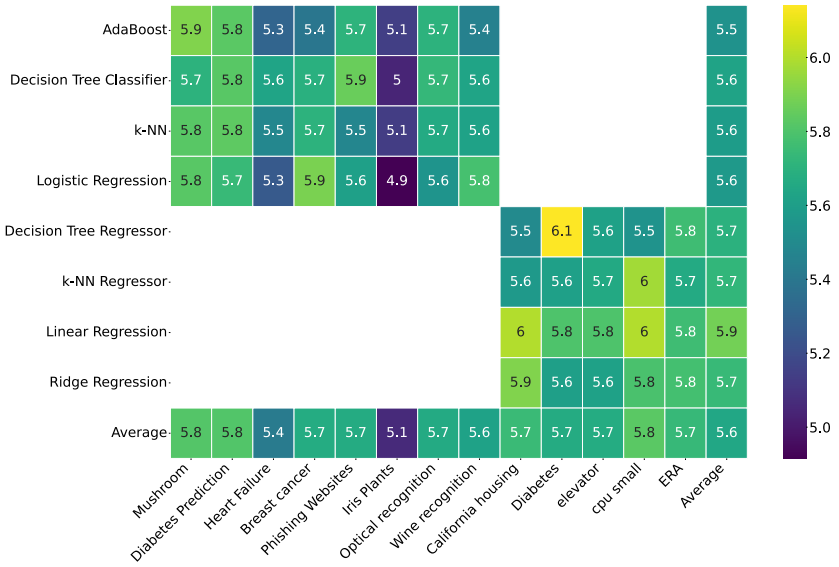

Figure 16: Heatmap of the average number of folds after e-fold cross-validation was stopped early in the hyperparameter optimatzation

It is important to highlight that the number of resources saved compared to the previous section 5.1 is even more significant. In total, 74150 hyperparameter combinations were evaluated with e-fold cross-validation as part of the hyperparameter optimization, which is almost ten times more than in section 5.1. Figure 17 shows the percentage distribution after how many folds the e-fold cross-validation stopped in these 74150 runs. The x-axis shows the number of folds after which e-fold cross-validation stopped. The y-axis shows the number of all iterations of the hyperparameter experiment. The results show that in 36.9% of cases, and therefore the clear majority, the e-fold cross-validation was stopped after 4 folds. In 18.5% after 5 and 17.9% after 6 folds. In only 4.7% of cases was early stopping not possible, which demonstrates that e-fold cross-validation also requires significantly fewer folds in hyperparameter optimization than 10-fold cross-validation.

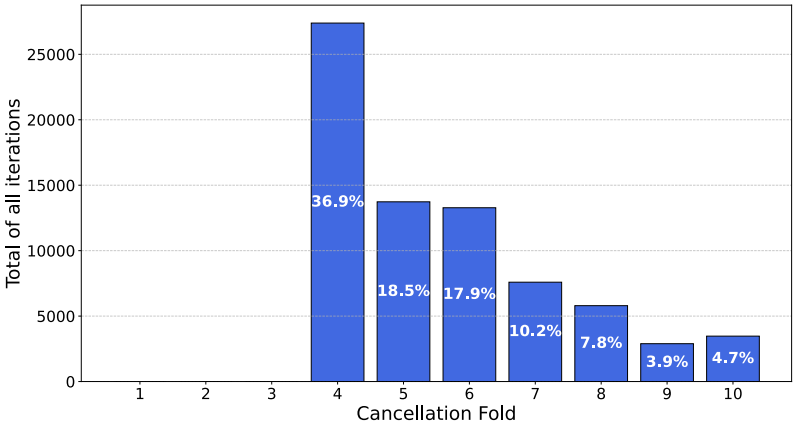

Figure 17: Distribution of the required folds of e-fold cross-validation in the hyperparameter optimization across the hyperparameter experiment

The findings above provide a solid foundation to answer the second key research question. E-fold cross-validation is conditionally suitable as an evaluation method for hyperparameter optimization.
Particularly noteworthy are the 4.4 folds saved on average compared to using hyperparameter optimization with the 10-fold cross-validation. At 55.5%, the hyperparameter optimization with e-fold cross-validation found more than half of the best hyperparameter combinations. The resulting 44.5% not equal best found hyperparameters show with less than 2% in the evaluation on the test data that they also deliver good comparable results.
This could indicate that although the best hyperparameter combination was not always found, the hyperparameter space was still analyzed as accurately as in the hyperparameter optimization with 10-fold cross-validation. A deeper analysis of how exactly the 44.5% non-optimally found hyperparameters deviate from the optimally identified hyperparameter combination would be useful here. Future research can take a closer look at this field. Due to the limited scope of this bachelor thesis, this field was not analyzed in more detail.
In certain applications where high accuracy is required, the choice of using e-fold cross-validation in the hyperparameter optimization may be worse than using 10-fold cross-validation, depending on the dataset, because the results have shown that there are large outliers in the percentage deviation on the test set, indicating that in some cases e-fold cross-validation may have stopped too early.

5.3 Impact of adjusting the hyperparameters of the e-fold cross-validation method

The values shown in Figure 18 and 19 figures refer to the average of all algorithm-dataset combinations. Specifically, a parameter change has the same scope as described in Section 5.1, but is not described in such detail here. Instead, it describes how the overall mean values for the percentage difference, the used folds and the number of iterations that fall within the confidence interval have changed as a result of the parameter adjustment.
Figure 18 shows the different results of the values used for the parameter e_{check}. The y-axis on the left-hand side shows the number of folds used and also the average deviation of the performance metric compared to the 10-fold cross-validation. The y-axis on the right-hand side describes the number of iterations that fall within the confidence interval. The x-axis shows the different values tested for e_{check}.

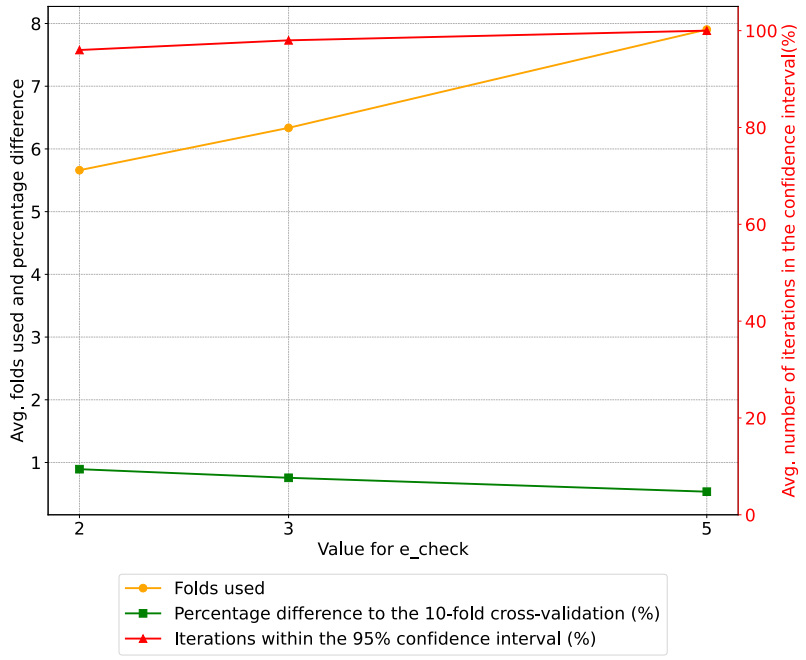

Figure 18: Impact of the adjustment of the hyperparameter e_{check} on the results of the e-fold cross-validation

As aspected, a higher value for e_{check} results in fewer folds being saved, but the model performance is closer to the model performance of the 10-fold cross-validation and the total number of iterations is more often within the confidence interval. Conversely, a lower value for e_{check} results in more saved folds, but the model performance is further away from the model performance of the 10-fold

cross-validation and the number of iterations within the confidence interval is lower. More precisely, the parameter $e_{check} = 2$ needs the fewest folds (5.7) but has the lowest percentage of iterations in the confidence interval (96.4%) and the largest deviation compared to 10-fold cross-validation (0.9%). In the hyperparameter combination used for this evaluation, these results are the same results as in 5.1. The diagram shows how the average results in 5.1 would have changed if the parameter on e_{check} had been set to 3 or 5. Furthermore, it can be observed that the impact of the different parameters influences the number of folds more than the confidence interval and the percentage deviation to 10-fold cross-validation.

Figure 19 shows the different results of the values used for the parameter e_{count}. The y-axes are identical to Figure 18. The x-axis shows the tested values for e_{count}. By selecting a higher value for e_{count}, fewer folds are saved, but the model performance is closer to the model performance of the 10-fold cross-validation and more iterations are within the confidence interval. Conversely, the smaller the value for e_{count}, the more folds are saved, but the model performance is further away from the model performance of the 10-fold cross-validation and fewer iterations are within the confidence interval.

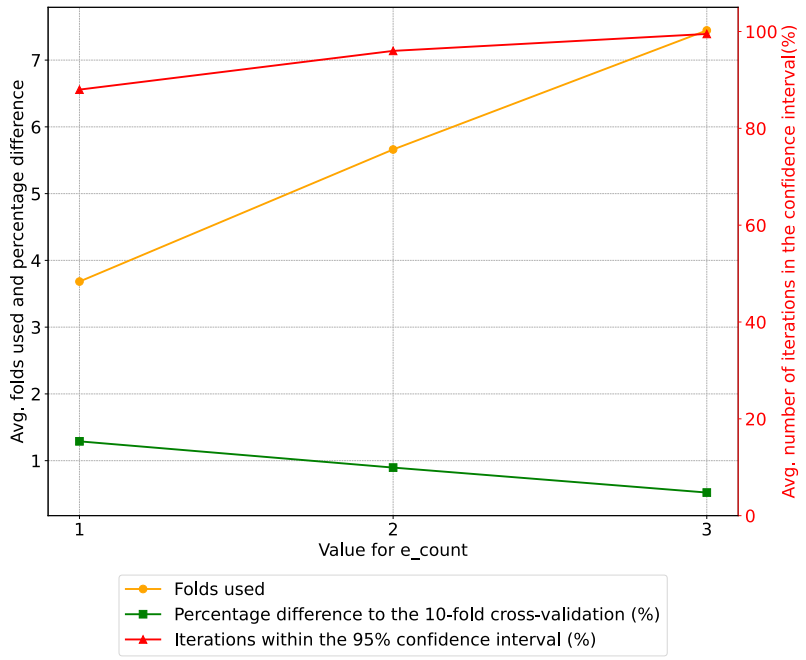

Figure 19: Impact of the adjustment of the hyperparameter e_{count} on the results of the e-fold cross-validation

The results of the $e_{count} = 2$ value again show the results from 5.1. The results show that in the evaluation in 5.1 significant more folds could have been saved by adjusting the parameter e_{count} from 2 to 1. To be more precise, with $e_{count} = 1$ on average 3.7 folds are required, which is the lowest of the used values. On the other hand, the percentage deviation of 1.3% compared to the 10-fold cross-validation is the highest deviation and, with 88% of all iterations within the confidence interval, the worst of all combinations. Similar to the previous Figure 18, choosing a higher value for e_{count} influences the number of saved folds more than the percentage deviation and the number of iterations in the confidence interval.

All in all, these results answer the third key research question. By optimizing the hyperparameters of e-fold cross-validation, the e-fold cross-validation method can effectively be further optimized. As expected, the optimization process has two directions: either save more folds and tolerate a higher deviation from the 10-fold cross-validation or save fewer folds and achieve a lower deviation from the 10-fold cross-validation. Depending on the situation and the goal, e-fold cross-validation can be optimized in a specific way.

6 Conclusion

In this thesis, the resource consumption of model validation in machine learning was reduced by developing and evaluating the e-fold cross-validation method. The goal of this thesis was to develop an energy-efficient alternative to k-fold cross-validation that still provides accurate results.
An analysis with 15 datasets and 10 algorithms was performed which were repeated 100 times with a different data distribution to show the robustness and generalizability of the method. The e-fold cross-validation was directly compared to the 10-fold cross-validation. By dynamically adjusting the number of folds, the consumption of resources was significantly reduced. E-fold cross-validation was able to save around 40% of the computing resources on average across all datasets and algorithms compared to 10-fold cross-validation. The percentage difference in performance metrics compared to the standard 10-fold cross-validation was often within a 0.5-2% range and is therefore minimal.
In addition, it was investigated whether the e-fold cross-validation method is suitable for hyperparameter optimization. For this investigation, 13 datasets from the 15 mentioned above and 8 of the 10 algorithms with different hyperparameter combinations were used. The results have shown that in the hyperparameter optimization with e-fold cross-validation, about 40% of the resource could be saved. On average, 55.5% of the best hyperparameter combinations were found using e-fold cross-validation compared to using 10-fold cross-validation for the hyperparameter optimization, which is more than half. Furthermore, the percentage deviation of the performance metric on the test score of the incorrectly detected hyperparameter combinations compared to the correctly detected ones, was less than 2%, which indicates that e-fold has also explored the hyperparameter space well for these combinations.
In addition, e-fold cross-validation is easy to implement and can be quickly applied to new and existing machine-learning projects. These results provide a good insight into the amount of energy and resources that can be saved and how e-fold cross-validation can save resources while still achieving good results that are close to the optimum.
The goal of this thesis was achieved, as e-fold cross-validation achieved significant savings in resources (approx. 40%) with minimal deviation in performance (0.9%) compared to traditional 10-fold cross-validation and also proved to be an effective method for hyperparameter optimization. The main success factor is the dynamic adjustment of the number of folds. Furthermore, the adaptation of the e-fold cross-validation method by its own hyperparameters should be emphasized. By adjusting the four hyperparameters e_{max}, e_{count}, $diff_{max}$ and e_{check} e-fold cross-validation can be adjusted in a way that either saves resources more effectively or is based more on achieving better performance and therefore uses more folds.
However, despite the achieved goal, critical aspects must be taken into account. Figure 4 has shown that there are cases in practice where e-fold cross-validation can stop too early, which can lead to an incorrect estimation of the model performance. It was also frequently found that smaller datasets performed worse

than larger ones. It would be interesting to analyze this correlation in more detail, possibly through deeper analyses of whether smaller datasets perform better if the hyperparameters of the e-fold cross-validation are selected in such a way that more folds are evaluated.
In addition, it would be useful to further investigate the optimization of hyperparameters using extended search spaces and alternative methods such as Bayesian optimization or random search. Future studies should investigate the difference between suboptimal and optimal hyperparameter combinations in the hyperparameter optimization with e-fold cross-validation in more detail to determine how much the hyperparameter spaces actually differ.
Despite the solid foundation on which this thesis is based, it is limited in the scope of datasets and algorithms tested. This scope can be extended by future work to include larger datasets and more complex models. Especially with larger datasets, it would be useful to examine whether and to what extent the e-fold cross-validation method is limited. A potential limitation could also be the additional calculation time and overhead of e-fold cross-validation, although this should not be significant.
This thesis opens several future research topics. Future studies can investigate the extension of e-fold cross-validation to other domains, such as deep learning or recommender systems. Another interesting approach for future work could be the comparison of e-fold cross-validation in hyperparameter optimization with Greedy cross-validation or early stopping although these approaches initially pursue a different goal, their approach could achieve a similar goal as e-fold cross-validation. Further research could also explore the impact of adjusting the e-fold cross-validation hyperparameter e_{max} to a higher or lower value.
To summarize, e-fold cross-validation is a promising way to achieve resource efficient and faster model evaluation without compromising the accuracy or reliability of the results.

References

[1] Stefan Aeberhard and M. Forina. Wine. UCI Machine Learning Repository, 1992. DOI: https://doi.org/10.24432/C5PC7J.

[2] D. Anguita, Luca Ghelardoni, Alessandro Ghio, L. Oneto, and Sandro Ridella. The 'k' in k-fold cross validation. In *The European Symposium on Artificial Neural Networks*, 2012.

[3] Jöran Beel, Lukas Wegmeth, and Tobias Vente. e-fold cross-validation: A computing and energy-efficient alternative to k-fold cross-validation with adaptive folds. June 2024. OSF Preprints.

[4] Daniel Belete and Manjaiah D H. Grid search in hyperparameter optimization of machine learning models for prediction of hiv/aids test results. *International Journal of Computers and Applications*, 44:1–12, 09 2021.

[5] Edward Bergman, Lennart Purucker, and Frank Hutter. Don't waste your time: Early stopping cross-validation, 2024.

[6] Matthias Feurer, Aaron Klein, Katharina Eggensperger, Jost Springenberg, Manuel Blum, and Frank Hutter. Efficient and robust automated machine learning. In *Advances in Neural Information Processing Systems 28 (2015)*, pages 2962–2970, 2015.

[7] R. A. Fisher. Iris. UCI Machine Learning Repository, 1936. DOI: https://doi.org/10.24432/C56C76.

[8] Haibo He and Edwardo A. Garcia. Learning from imbalanced data. *IEEE Transactions on Knowledge and Data Engineering*, 21(9):1263–1284, 2009.

[9] Urwah Imran, Asim Waris, Maham Nayab, and Uzma Shafiq. 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)*, pages 1–4, 2023.

[10] Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani, and Jonathan Taylor. *An Introduction to Statistical Learning*. Springer, 2nd edition, 2023.

[11] Kevin Jamieson and Ameet Talwalkar. Non-stochastic best arm identification and hyperparameter optimization. In Arthur Gretton and Christian C. Robert, editors, *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, volume 51 of *Proceedings of Machine Learning Research*, pages 240–248, Cadiz, Spain, 09–11 May 2016. PMLR.

[12] Roger W. Johnson. Cholesterol levels data. https://www.openml.org/d/560, 2024. Accessed via OpenML on 2024-09-15.

[13] Ömer Karal. Performance comparison of different kernel functions in svm for different k value in k-fold cross-validation. In *2020 Innovations in Intelligent Systems and Applications Conference (ASYU)*, pages 1–5, 2020.

[14] Rabie El Kharoua. Air quality and health impact dataset, 2024.

[15] Rabie El Kharoua. Students performance dataset, 2024.

[16] Ron Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'95, page 1137–1143, San Francisco, CA, USA, 1995. Morgan Kaufmann Publishers Inc.

[17] Alexandre Lacoste, Alexandra Luccioni, Victor Schmidt, and Thomas Dandres. Quantifying the carbon emissions of machine learning, 2019.

[18] Giovanni Di Leo and Francesco Sardanelli. Statistical significance: p value, 0.05 threshold, and applications to radiomics—reasons for a conservative approach. *European Radiology Experimental*, 4(1):18, 2020.

[19] Sekander Hayat Khan M. *Standard Deviation*, pages 1378–1379. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

[20] Christopher Mahlich. Open source project for e-fold cross-validation. https://code.isg.beel.org/e-fold-ml-mahlich/tree/Bachelor-thesis, 2024. GitHub repository.

[21] Bruce G. Marcot and Anca M. Hanea. What is an optimal value of k in k-fold cross-validation in discrete bayesian network analysis? *Computational Statistics*, 36(3):2009–2031, 2021.

[22] Rami Mohammad and Lee McCluskey. Phishing Websites. UCI Machine Learning Repository, 2012. DOI: https://doi.org/10.24432/C51W2X.

[23] Osval Montesinos-López, Abelardo Montesinos, and Jose Crossa. *Multivariate Statistical Machine Learning Methods for Genomic Prediction.*

[24] Mustafa Mustafa. Diabetes prediction dataset. https://www.kaggle.com/datasets/iammustafatz/diabetes-prediction-dataset, 2023. Accessed: 2024-09-15.

[25] Isaac Nti, Owusu Nyarko-Boateng, and Justice Aning. Performance of machine learning algorithms with different k values in k-fold cross-validation. *International Journal of Information Technology and Computer Science*, 6:61–71, 12 2021.

[26] Opeoluwa Oyedele. Determining the optimal number of folds to use in a k-fold cross-validation: A neural network classification experiment. *Research in Mathematics*, 10(1):2201015, 2023.

[27] David Patterson, Joseph Gonzalez, Urs Hölzle, Quoc Le, Chen Liang, Lluis-Miquel Munguia, Daniel Rothchild, David R. So, Maud Texier, and Jeff Dean. The carbon footprint of machine learning training will plateau, then shrink. *Computer*, 55(7):18–28, 2022.

[28] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[29] Wolfgang Roth, Günther Schindler, Bernhard Klein, Robert Peharz, Sebastian Tschiatschek, Holger Fröning, Franz Pernkopf, and Zoubin Ghahramani. Resource-efficient neural networks for embedded systems, 2024.

[30] Roy Schwartz, Jesse Dodge, Noah A. Smith, and Oren Etzioni. Green ai, 2019.

[31] scikit-learn. California housing dataset, 2024. Author not specified, accessed via scikit-learn on 2024-09-15.

[32] scikit-learn. Diabetes dataset, 2024. Author not specified, accessed via scikit-learn on 2024-09-15.

[33] scikit-learn. Digits dataset, 2024. Author not specified, accessed via scikit-learn on 2024-09-15.

[34] Michael Smithson. *Confidence Interval*, pages 283–284. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

[35] Daniel Soper. Hyperparameter optimization using successive halving with greedy cross validation. *Algorithms*, 16:17, 12 2022.

[36] Daniel S. Soper. Greed is good: Rapid hyperparameter optimization and model selection using greedy k-fold cross validation. *Electronics*, 10(16), 2021.

[37] Emma Strubell, Ananya Ganesh, and Andrew McCallum. Energy and policy considerations for deep learning in nlp, 2019.

[38] Chris Thornton, Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. Auto-weka: Combined selection and hyperparameter optimization of classification algorithms, 2013.

[39] Jan van Rijn. Us census (1990) data. https://www.openml.org/d/216, 2024. Accessed via OpenML on 2024-09-15.

[40] Vanschoren, Joaquin. Mnist original. https://www.openml.org/d/562, 2024. Accessed via OpenML on 2024-09-15.

[41] Aadarsh Velu. Heart failure prediction clinical records dataset. https://www.kaggle.com/datasets/aadarshvelu/heart-failure-prediction-clinical-records, 2023. Accessed: 2024-09-15.

[42] Tobias Vente, Lukas Wegmeth, Alan Said, and Joeran Beel. From clicks to carbon: The environmental toll of recommender systems. In *Proceedings of the 18th ACM Conference on Recommender Systems*, 2024.

[43] Roberto Verdecchia, June Sallou, and Luís Cruz. A systematic review of green ai. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 13, 2023.

[44] Vijay Kumar Verma, Kanak Saxena, and Umesh Banodha. Analysis effect of k values used in k fold cross validation for enhancing performance of machine learning model with decision tree. In Deepak Garg, Joel J. P. C. Rodrigues, Suneet Kumar Gupta, Xiaochun Cheng, Pushpender Sarao, and Govind Singh Patel, editors, *Advanced Computing*, pages 374–396, Cham, 2024. Springer Nature Switzerland.

[45] Heider D. Wagner, Dennis and Georges Hattab. Secondary Mushroom. UCI Machine Learning Repository, 2021. DOI: https://doi.org/10.24432/C5FP5Q.

[46] Cort J. Willmott and Kenji Matsuura. Advantages of the mean absolute error (mae) over the root mean square error (rmse) in assessing average model performance. *Climate Research*, 30(1):79–82, 2005.

[47] Mangasarian Olvi Street Nick Wolberg, William and W. Street. Breast Cancer Wisconsin (Diagnostic). UCI Machine Learning Repository, 1993. DOI: https://doi.org/10.24432/C5DW2B.

[48] Tzu-Tsung Wong and Po-Yang Yeh. Reliable accuracy estimates from k-fold cross validation. *IEEE Transactions on Knowledge and Data Engineering*, 32(8):1586–1594, 2020.

[49] Sanjay Yadav and Sanyam Shukla. Analysis of k-fold cross-validation over hold-out validation on colossal datasets for quality classification. In *2016 IEEE 6th International Conference on Advanced Computing (IACC)*, pages 78–83, 2016.

Eidesstattliche Versicherung

Ich versichere, meine Arbeit (bei einer Gruppenarbeit meinen entsprechend gekennzeichneten Anteil der Arbeit) selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie Zitate kenntlich gemacht zu haben.

Alle Stellen, die dem Wortlaut oder dem Sinn nach (inkl. Übersetzungen) anderen Werken entnommen sind, habe ich in jedem einzelnen Fall unter genauer Angabe der Quelle (einschließlich des World Wide Web sowie anderer elektronischer Datensammlungen) deutlich als Entlehnung kenntlich gemacht. Dies gilt auch für angefügte Zeichnungen, bildliche Darstellungen, Skizzen und dergleichen. Ich nehme zur Kenntnis, dass die nachgewiesene Unterlassung der Herkunftsangabe als versuchte Täuschung gewertet wird.

Pattenberg 23.09.2024	
Ort, Datum	Unterschrift