cc

UNIKASSEL
VERSITÄT

Internship

Advancing Ensemble Regression: Automated Hyperparameter Tuning of the Diversity–Accuracy Trade-Off in Sign Diversity Metric

Samet Atak

Enrollment Number:	36336210
First Evaluator:	Univ.-Prof. Dr.-Ing. Andreas Kroll
Supervisor:	M.Sc. Farzad Rezzazadeh Pilehbarboni
Submission Date:	15/09/2025

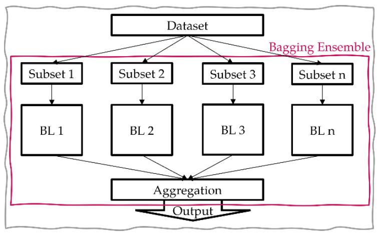
Mess- und Regelungstechnik
Univ.-Prof. Dr.-Ing. Andreas Kroll

Internship

Advancing Ensemble Regression: Automated Hyperparameter Tuning of Ensemble Size and the Diversity–Accuracy Trade-Off in Sign Diversity Metric

Samet Atak

The rapid advancements in data-driven modeling have made machine learning (ML) essential for predicting complex systems in production and quality assurance. However, a significant challenge arises when working with high-dimensional datasets that are also small in size: the risk of overfitting, where models perform well on training data but fail to generalize to new, unseen data. Ensemble learning offers a powerful solution to this issue by combining the predictions of multiple base learners (BLs), each exploring the data from a different perspective. This diversity enables the ensemble to achieve better overall performance than any individual model [1].



We are offering an internship opportunity focused on enhancing ensemble-learning techniques for regression tasks. The project builds on a previous student's work that developed a novel metric (Sign Diversity) for BL selection in ensemble learning [2, 3]. This work can be summarized in two phases. In the first phase, the Sign Diversity metric for selecting BLs in ensemble learning—balancing diversity and accuracy—was developed and integrated into a passive selection process: each BL was trained, and the developed metric was used to select the best combination of them. Various values of the trade-off parameter α—balancing diversity and accuracy—and different ensemble sizes were tested, and the results were documented. Phase 1 is fully complete.

In Phase 2, the existing code from Phase 1 was transferred into an optimization framework, in which each BL's hyperparameters were optimized via k-fold cross-validation within a grid-search process to find both α and the ensemble size.

The goal of this internship is to transition the existing Python code for both phases from k-fold cross-validation to leave-one-out cross-validation, which is preferred to deal with small -sized datasets. Additionally, instead of a sequential grid search over ensemble sizes and α values in the existing optimization framework, these parameters should be treated as hyperparameters and optimized using an algorithm such as Bayesian optimization—**providing a more efficient, automated approach and reducing the risk of becoming stuck in local minima**.

The ideal candidate should have a background in ML, proficiency in programming (Python), and a keen interest in ensemble learning. This project offers a unique opportunity to contribute to cutting-edge research in ML, with practical implications for predictive modeling in complex systems.

The work includes the following tasks:

- Develop a deep familiarity with ensemble learning and regression tasks in ML.
- Convert the existing Phase 1 and Phase 2 codebases from k-fold cross-validation to leave-one-out cross-validation, ensuring both pipelines fully support this validation method.
- Integrate the novel BL selection metric into an optimization framework and automate the tuning of ensemble size and α as hyperparameters via Bayesian optimization (or a similar technique).
- Apply the developed frameworks to available high-dimensional datasets, including the UHPC and 3MA datasets and at least two other test functions with high dimensionality, to evaluate the performance of the proposed ensemble structure.
- Documentation of work and colloquium presentation.

Supervisor: F. Rezazadeh M.Sc., Univ.-Prof. Dr.-Ing. A. Kroll

Start: 15.05.2025
End: 30.09.2025

References:

[1] Sagi, O., & Rokach, L. „Ensemble learning: A survey“. In: Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 8(4), e1249. 2018.

[2] Olfatbakhsh, E. “Improving Ensemble Regression Accuracy by Developing a New Base-Learner Selection Metric and Enhancing Diversity During the Learning Process”. Master’s thesis, University of Kassel. 2023.

[3] Rezazadeh, F., Olfatbakhsh, E., Kroll, A. „Sign Diversity: A Method for Measuring Diversity in Base Learner Selection for Ensemble Regression“. In: IEEE Symposium Series on Computational Intelligence (SSCI). 2025.

[4] Shahhosseini, M., Hu, G., & Pham, H. „Optimizing ensemble weights and hyperparameters of machine learning models for regression problems“. In: Machine Learning with Applications, 7, 100251. 2022.

[5] Olfatbakhsh, E. “Improving Ensemble Regression Accuracy by Developing a New Base-Learner Selection Metric and Enhancing Diversity During the Learning Process”. Master’s thesis, University of Kassel. 2023.

[6] Dong, X., Yu, Z., Cao, W., Shi, Y., & Ma, Q. “A survey on ensemble learning”. In: Frontiers of Computer Science, 14(2), 241-258. 2020.

[7] Palaniswamy, S. K., & Venkatesan, R. „ Hyperparameters tuning of ensemble model for software effort estimation “. In: Journal of Ambient Intelligence and Humanized Computing, 12(6), 6579-6589. 2021.

[8] Purdy, D. G. “Sparse Models for Sparse Data: Methods, Limitations, Visualizations and Ensembles”. Dissertation: https://escholarship.org/uc/item/9qb472v2, University of California, Berkeley. 2012.

[9] Błaszczyński, J., Stefanowski, J., & Słowiński, R. “Consistency driven feature subspace aggregating for ordinal classification”. In: International joint conference on rough sets, 580-589. 2016.

Declaration

I hereby declare that I have composed this work independently and have not used any sources or aids other than those specified.

Kassel , 15.09.2025 [illegible]

Ort, Datum, Unterschrift

Abstract

Stacking-Ensembles werden im Machine Learning häufig eingesetzt, da sie heterogene Basislerner (BLs) kombinieren und so die Vorhersageleistung verbessern können. Eine zentrale Herausforderung beim Aufbau von Ensembles liegt in der Wahl der geeigneten Ensemblegröße sowie im Ausgleich zwischen Genauigkeit und Diversität der BLs. Klassische Ansätze stützen sich oft auf feste Diversitätsmaße und aufwändige Gitter-Suchen, die sowohl rechenintensiv sind als auch für kleine, hochdimensionale Datensätze ungeeignet.

In dieser Arbeit wird die Weighted Sign Diversity Metric zur Extended Weighted Sign Diversity Metric (EWSDM) erweitert, wobei ein einstellbarer Parameter α eingeführt wird, der das Verhältnis zwischen Genauigkeit und Diversität steuert. Darüber hinaus wird ein automatisiertes Optimierungsframework vorgeschlagen, in dem die Hyperparameter der BLs, die Ensemblegröße und α gemeinsam mithilfe von Bayes'scher Optimierung abgestimmt werden. Um eine robuste Bewertung bei begrenzten Daten zu gewährleisten, verwendet das Framework Leave-One-Out-Cross-Validation (LOOCV) anstelle der herkömmlichen k-fachen Validierung.

Das Framework wird sowohl auf Benchmark-Funktionen als auch auf hochdimensionalen Datensätzen, darunter UHPC und 3MA, evaluiert. Die Ergebnisse zeigen, dass Ensembles, die durch EWSDM gesteuert werden, diejenigen übertreffen, die mit klassischen Diversitätsmaßen wie Q-Statistik, Korrelation und Disagreement konstruiert wurden. Darüber hinaus liefert die Bayes'sche Optimierung kompakte und effektive Ensembles, verringert das Risiko des Overfittings und verbessert die Recheneffizienz.

Diese Arbeit verdeutlicht das Potenzial, fortgeschrittene Diversitätsmaße mit probabilistischer Hyperparameter-Optimierung zu kombinieren, um genauere, zuverlässigere und effizientere Stacking-Ensembles für Regressionsaufgaben zu entwickeln.

Contents

List of Figures

1 Introduction

In the modern era, data has become a central resource for decision-making across industries and scientific domains. Instead of relying on intuition or past experience, organizations now increasingly depend on the systematic analysis of data. This shift has led to remarkable improvements in efficiency, accuracy, and transparency. At the same time, however, it has introduced new challenges, particularly in areas where systems are complex, dynamic, and influenced by many interacting factors.

Production and quality assurance are examples of such fields, where outcomes are shaped by variables that are often difficult to capture or predict. The unpredictability of these systems makes modeling a demanding task. The difficulty becomes even greater when the amount of data available for training is limited. Advanced machine learning algorithms typically require large datasets to function well; with smaller datasets, they may underperform, fail to reflect the variability of the system, and produce unreliable predictions.

To address these difficulties, researchers and practitioners have turned toward ensemble learning techniques. These approaches combine multiple algorithms, with the idea that their collective knowledge can deliver stronger and more stable results than any single model on its own. Among the various strategies, stacking stands out for its ability to integrate diverse base learners in a layered structure, enabling the ensemble to capture different aspects of the data and achieve higher predictive accuracy.

Nevertheless, stacking is not without its own challenges. Balancing the trade-off between accuracy and diversity among the base learners is a complex task. Selecting suitable models and integrating their outputs through a meta-learner add further layers of complexity. Without careful design, these factors can limit the effectiveness of the ensemble.

The present work aims to investigate these challenges in depth and propose improvements to stacking ensemble regression. In particular, the study explores the use of different cross-validation techniques to strengthen the robustness of results. At the same time, it addresses the issue of heavy computational demands by developing solutions to accelerate training on multi-core systems. Through these contributions, the work seeks to make stacking ensembles not only more accurate but also more practical and efficient in real-world applications.

2 Ensemble Learning

2.1 Fundamentals of Machine Learning

2.1.1 Basic Concepts of Machine Learning

Machine learning (ML) is a subfield of artificial intelligence that focuses on algorithms capable of learning from data and improving performance without explicit programming [2]. Unlike traditional rule-based approaches, ML models adapt as they encounter more data, enabling them to generalize patterns and make predictions in complex, dynamic environments. This adaptability is what makes ML central to applications such as image recognition, natural language processing, recommendation systems, and autonomous driving.

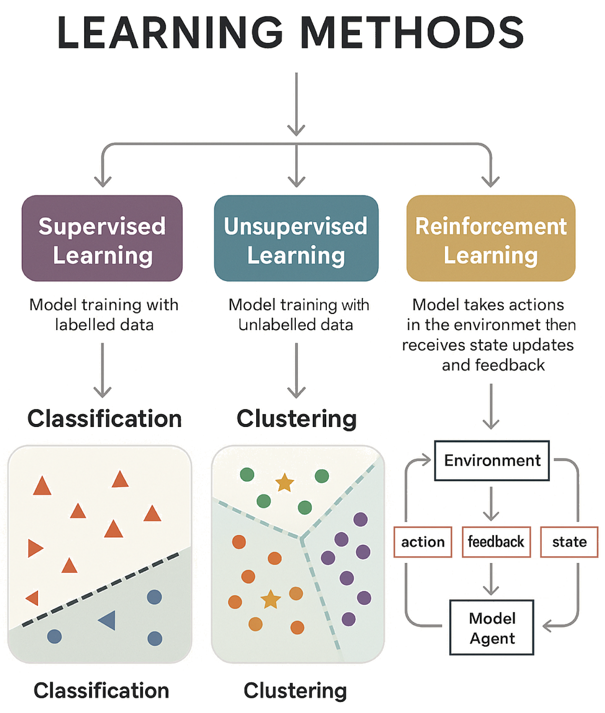
At its core, ML relies on three essential components: data, models, and learning algorithms. Data provides the raw observations from which patterns are inferred. Models serve as mathematical or probabilistic structures that capture these patterns, ranging from simple linear functions to deep neural networks. Learning algorithms optimize model parameters to minimize errors, typically by solving an optimization problem. In formal terms, the goal is to approximate an unknown function $f(x)$ by minimizing a loss function $L(f(x), y)$, where y is the true label or outcome [3]. The choice of loss function—such as mean squared error in regression or cross-entropy in classification—plays a crucial role in guiding the learning process.

One of the central challenges in ML is the trade-off between underfitting and overfitting. Underfitting occurs when a model is too simplistic to capture meaningful relationships in the data, resulting in poor performance on both training and test sets. Overfitting arises when the model becomes overly complex, memorizing training data—including noise—at the expense of generalization. To mitigate these issues, practitioners use techniques like cross-validation, regularization, early stopping, and ensemble learning, all of which help build models that generalize more effectively to unseen data.

Machine learning is commonly divided into three paradigms: supervised learning, where models are trained on labeled data to predict outputs; unsupervised learning, which identifies structures and clusters in unlabeled data; and reinforcement learning, where agents interact with an environment and learn through rewards and penalties [4]. These paradigms provide the conceptual backbone of ML and form the basis for advanced approaches such as semi-supervised and self-supervised learning. Together, they create a flexible framework for tackling diverse real-world problems across science, industry, and everyday technology.

2.1.2 Learning Methods

Machine learning is commonly divided into three major learning paradigms: supervised learning, unsupervised learning, and reinforcement learning. Each method differs in how data and feedback are provided to the model.



Supervised Learning
Supervised learning is an ML technique that relies on labeled data. Each input is paired with a corresponding output, and the model learns a mapping function between them. Typical applications include classification (e.g., image recognition) and regression (e.g., predicting house prices) [2].

Unsupervised Learning
In unsupervised learning, the data does not contain explicit labels. The goal is to identify hidden patterns, clusters, or structures within the dataset. Common techniques include clustering (e.g., k-means) and dimensionality reduction (e.g., PCA) [2].

Reinforcement Learning
Reinforcement learning is based on an agent–environment interaction. The agent learns by performing actions and receiving rewards or penalties, with the objective of maximizing long-term cumulative reward. This paradigm is widely applied in robotics, autonomous driving, and game playing [4].

2.1.2 Base Learners

In ensemble learning, **base learners** refer to the individual predictive models that form the building blocks of an ensemble. Each base learner is trained on the data and makes its own predictions. By combining these diverse models, the ensemble can reduce variance, control bias, and achieve better generalization [5, 6]. The following subsections describe the base learners employed in this study. For each method, its mathematical formulation is presented along with an explanation of the involved parameters.

Linear Regression

Linear Regression (LR) is one of the earliest and most widely used statistical learning methods, dating back to classical regression theory but later formalized in the context of machine learning [2]. It assumes a linear relationship between the predictors and the response variable, expressed as:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p + \epsilon \tag{2.1}$$

where β_j are the coefficients reflecting the contribution of each feature, β_0 is the intercept, and ϵ denotes random noise. The estimation of these coefficients relies on minimizing the residual sum of squares, a principle that remains central in statistical modeling [6].

Lasso Regression

While ordinary least squares regression may suffer from overfitting in high-dimensional settings, Lasso Regression introduces an L_1 penalty on the coefficients, encouraging sparsity [6]. This not only shrinks parameters but also performs implicit feature selection, making it suitable for problems with many correlated variables. Its optimization problem is written as:

$$\hat{\beta} = \arg\min_{\beta} \left\{ \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^{p} |\beta_j| \right\} \tag{2.3}$$

Here, λ is the regularization strength, where large values eliminate irrelevant predictors while smaller values allow the model to remain flexible.

Kernel Ridge Regression

Kernel Ridge Regression (KRR) extends ridge regression by leveraging the kernel trick, thus enabling non-linear relationships to be modeled in high-dimensional feature spaces [3]. Unlike ordinary linear regression, the hypothesis function is expressed through kernels that measure similarity between data points:

$$f(x) = \sum_{i=1}^{n} \alpha_i K(x_i, x) \tag{2.5}$$

where $K(x_i, x)$ can be chosen as a linear, polynomial, or radial basis function kernel. The regularization parameter λ prevents overfitting by penalizing overly complex functions, ensuring smoother solutions [2].

Decision Trees

Decision Trees (DTs) represent a class of non-parametric learners that partition the input space into hierarchical regions by recursively splitting on feature values [6]. Each split is selected to minimize prediction error, typically measured by squared error for regression:

$$\min_{j,s} \left[\sum_{i \in R_1(j,s)} (y_i - \hat{y}_{R_1})^2 + \sum_{i \in R_2(j,s)} (y_i - \hat{y}_{R_2})^2 \right] \tag{2.6}$$

Here, j denotes the feature chosen for splitting, s the split threshold, and $\hat{y}_{R_1}, \hat{y}_{R_2}$ are the average responses in the resulting regions R_1 and R_2. The tree grows by selecting splits that minimize prediction error, balancing accuracy with generalization.

K-Nearest Neighbors

K-Nearest Neighbors (KNN) is an instance-based learning method that makes predictions by averaging the outcomes of the k closest samples in the training data [7]. The underlying assumption is that points close to each other in feature space are likely to share similar outputs.

$$\hat{f}(x) = \frac{1}{k} \sum_{x_i \in \mathcal{N}_k(x)} y_i \tag{2.7}$$

Here, $\mathcal{N}_k(x)$ is the set of nearest neighbors of x. The choice of k is crucial: small values may capture noise, whereas large values may oversmooth predictions. Despite its simplicity, KNN remains competitive for tasks such as pattern recognition and recommendation systems [6].

Gaussian Process Regression

Gaussian Process Regression (GPR) is a Bayesian, non-parametric method that places a probability distribution over functions, rather than assuming a fixed parametric form [8]. This allows the model to provide not only point estimates but also principled uncertainty measures. The posterior predictive distribution for a new input x_* is given by:

$$f(x_*) \mid X, y, x_* \sim \mathcal{N}(\bar{f}_*, \, \mathrm{cov}(f_*)) \tag{2.8}$$

The mean $\bar{f}_*$ and covariance $\mathrm{cov}(f_*)$ are derived from kernel functions, which encode similarity between data points. Common kernels include the Radial Basis Function (RBF) and Matérn kernels, which allow practitioners to incorporate prior beliefs about smoothness or periodicity. Although computationally expensive for large datasets, GPR is especially valued in scientific applications where uncertainty quantification is as important as prediction accuracy.

Support Vector Regression

Support Vector Regression (SVR) extends the principles of Support Vector Machines (SVMs) to regression tasks [9]. Instead of aiming for perfect prediction of every data point, SVR defines an ϵ-insensitive tube around the target function, ignoring small deviations while penalizing larger ones. This makes it robust against noise and outliers.

$$(\hat{\beta}, \hat{\beta}_0) = \arg\min_{\beta, \beta_0} \left(\frac{1}{2} \|\beta\|^2 + C \sum_{i=1}^{n} L_\epsilon(y_i - f(x_i)) \right) \tag{2.9}$$

Here, the first term $\frac{1}{2}\|\beta\|^2$ enforces flatness of the function, while the second term introduces penalties for errors larger than ϵ. The parameter C balances model complexity against tolerance for errors. Combined with kernel functions, SVR can effectively model non-linear relationships in high-dimensional spaces [10].

Random Forest

Random Forest (RF) is an ensemble method that aggregates multiple decision trees, each trained on bootstrap samples with randomized feature subsets [11]. This combination reduces variance and mitigates the overfitting typically observed in single decision trees. The prediction is computed as the average of individual tree outputs:

$$\hat{f}(x) = \frac{1}{B} \sum_{b=1}^{B} f_b(x) \tag{2.13}$$

where B is the number of trees. RF has become a benchmark algorithm due to its robustness, ease of use, and strong empirical performance across diverse applications such as bioinformatics, finance, and text classification.

Gradient Boosting

Gradient Boosting (GB) builds an ensemble of weak learners, typically shallow decision trees, in a sequential manner. Each new tree is trained to correct the residuals of the previous ensemble [12]. The iterative update is given by:

$$F_m(x) = F_{m-1}(x) + \nu \cdot h_m(x) \tag{2.14}$$

Here, $h_m(x)$ is the weak learner trained on residuals, and ν is the learning rate controlling its contribution. Small values of ν often improve generalization but require more trees. Gradient Boosting has been the foundation for state-of-the-art implementations such as XGBoost and LightGBM, which dominate machine learning competitions.

Partial Least Squares

Partial Least Squares (PLS) regression is a dimensionality reduction and regression method that projects predictors and responses into a shared latent space. Unlike principal component regression, PLS explicitly searches for components that maximize the covariance between predictors and the response [13].

$$\max_{w,c} \ \text{Cov}(Xw, yc) \tag{2.15}$$

Here, Xw denotes the projection of predictors and yc the projection of the response. By focusing on maximizing covariance, PLS is especially suited for datasets with many correlated features, such as those encountered in chemometrics and genomics.

2.1.3 Hyperparameters in Machine Learning Models

In machine learning, the performance of an algorithm is determined not only by the learning process and the available data, but also by a set of external configuration variables known as *hyperparameters*. Unlike model parameters, which are learned directly from the data during training (e.g., regression coefficients or neural network weights), hyperparameters are defined before the learning process begins and control the behavior of the algorithm itself [2].

The proper configuration of hyperparameters is crucial, as they directly affect both the predictive accuracy and the generalization ability of a model. Poorly chosen hyperparameters can lead to underfitting (an overly simplistic model that fails to capture patterns in the data) or overfitting (a model that memorizes noise in the training data rather than generalizable structures). Therefore, hyperparameters play a central role in balancing the bias–variance trade-off, one of the fundamental challenges in machine learning [3].

Another important distinction is that hyperparameters are often specific to the algorithmic family. For instance, linear models typically involve only a few regularization-related hyperparameters, whereas ensemble methods such as Gradient Boosting or Random Forests contain many interacting hyperparameters related to the number of estimators, tree depth, and learning rate. This diversity makes the process of selecting hyperparameters non-trivial, especially in ensemble learning where multiple base learners with heterogeneous configurations are combined.

As a result, hyperparameter tuning has become an essential step in machine learning pipelines. Properly adjusted hyperparameters not only improve predictive performance but also enhance computational efficiency and robustness. This motivates the use of systematic optimization methods such as grid search, random search, and Bayesian optimization [14].

2.1.4 Hyperparameter Optimization Methods

In machine learning, the predictive performance of a model is not only determined by the learning algorithm itself but also by the choice of *hyperparameters*. Hyperparameters are external configuration settings that control how a model is trained, such as the learning rate in gradient boosting, the maximum depth of a decision tree, or the regularization strength in regression models. Choosing the right hyperparameters is critical because they directly affect model complexity, generalization ability, and computational efficiency. For instance, poorly tuned hyperparameters can lead to underfitting (a model too simple to capture patterns) or overfitting (a model too complex, fitting noise instead of patterns) [2].

Hyperparameter optimization (HPO) is the process of systematically searching for the best hyperparameter configuration for a given model and dataset. Instead of relying on intuition or trial-and-error, HPO provides structured methods to explore the hyperparameter space and find values that maximize performance according to a chosen evaluation metric (e.g., accuracy, RMSE, or R^2). The optimization process typically involves three components: (1) a search space that defines the possible values of each hyperparameter, (2) a search strategy that decides which hyperparameters to evaluate, and (3) an evaluation function that measures performance, usually via cross-validation.

Several strategies have been proposed for hyperparameter optimization, ranging from simple exhaustive methods to advanced probabilistic approaches. The most commonly used methods include Grid Search, Random Search, and Bayesian Optimization.

Grid Search
Grid search is the most straightforward method of hyperparameter optimization. In this approach, the user defines a finite set of values for each hyperparameter, and the algorithm evaluates the model for all possible combinations of these values. For example, if the learning rate is set to $\{0.01, 0.05, 0.1\}$ and the maximum depth of a tree is set to $\{3, 5, 7\}$, grid search will evaluate the model nine times—once for each pair of values. The advantage of grid search is its simplicity and completeness: it is guaranteed to test every combination within the grid, so the best configuration in that space will be found. However, grid search suffers from the *curse of dimensionality*. As the number of hyperparameters or the number of candidate values increases, the number of required evaluations grows exponentially, making the method computationally very expensive [15].

Random Search
Random search provides a more efficient alternative by sampling hyperparameter values at random from predefined distributions. Instead of exhaustively testing every combination, it randomly selects a subset of possible configurations. Surprisingly, research has shown that random search often outperforms grid search in practice, especially when only a small number of hyperparameters significantly influence model performance [14]. For example, if only the learning rate has a large effect while other parameters are less important, random search is more likely to explore a wide range of learning rates rather than wasting time on unimportant combinations. Random search is also highly parallelizable and can

handle both discrete and continuous hyperparameter spaces efficiently. Its main limitation is that it does not explicitly use information from previous evaluations—each trial is independent.

Bayesian Optimization

Bayesian optimization is a more advanced strategy that addresses the inefficiency of random and grid search. Instead of treating each evaluation as independent, Bayesian optimization builds a probabilistic model (called a *surrogate model*) of the objective function that maps hyperparameters to model performance. A common surrogate is the Gaussian Process, which not only predicts the expected performance of a new configuration but also estimates the uncertainty of that prediction [1]. An *acquisition function* is then used to decide which hyperparameters to test next, balancing *exploration* (trying uncertain regions of the search space) and *exploitation* (focusing on promising regions already identified). This iterative process allows Bayesian optimization to find near-optimal hyperparameters with far fewer evaluations compared to grid or random search.

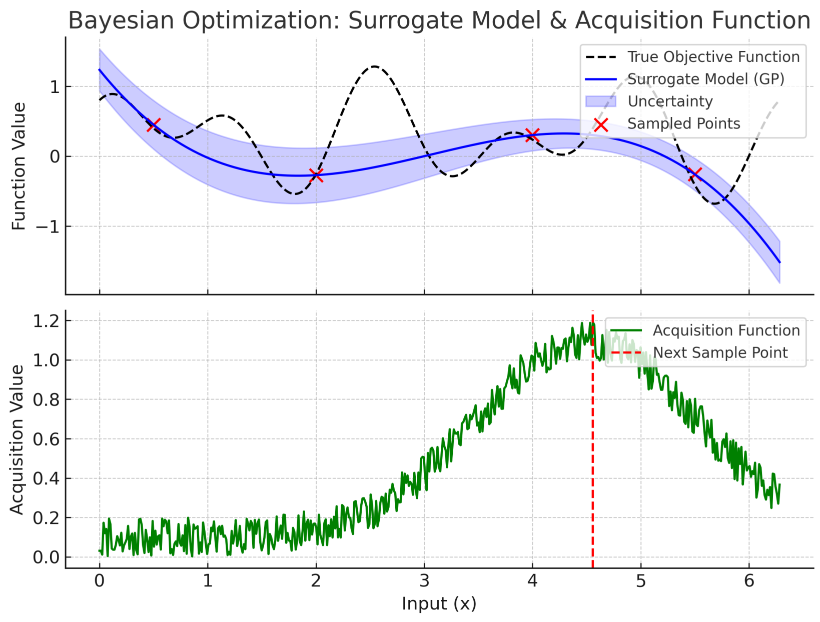


Figure 2.1: Illustration of Bayesian Optimization, where a surrogate model and acquisition function guide the search for optimal hyperparameters [1].

Figure 2.1 illustrates the Bayesian optimization process. The surrogate model approximates the true performance function, and the acquisition function guides the search towards areas with the highest potential improvement. As more evaluations are performed, the surrogate becomes more accurate, and the algorithm converges toward the best hyperparameter configuration. Because of its efficiency, Bayesian optimization has become the method of choice in many modern machine learning workflows, especially when model training is computationally expensive.

In summary, grid search offers completeness but quickly becomes infeasible for large search spaces, random search is simple and often surprisingly effective, and Bayesian optimization provides an intelligent balance between efficiency and accuracy. As a result, Bayesian optimization has emerged as one of the most powerful approaches for hyperparameter tuning in complex machine learning models [16].

2.2 Introduction to Ensemble Learning

Ensemble learning is a paradigm in machine learning where multiple individual models, known as base learners, are combined to produce a stronger and more accurate predictive model [17]. The key intuition is that while a single model may be prone to errors due to bias or variance, an ensemble can mitigate these issues by aggregating the strengths of its members. This approach is grounded in the statistical principle that averaging multiple independent predictions tends to reduce error variance and enhance generalization performance.

Ensemble methods have become central to modern machine learning practice. They are used in both classification and regression tasks, and they frequently outperform single models in terms of accuracy and robustness [18]. Prominent examples include bagging, boosting, and stacking, which differ in how base learners are trained and combined. Bagging reduces variance by averaging predictions from models trained on bootstrap samples, boosting reduces bias by sequentially improving weak learners, and stacking integrates diverse models using a meta-learner to optimize final predictions [6].

Beyond performance improvements, ensemble learning also enhances model stability and robustness in the presence of noisy data. This has led to their adoption in diverse applications, from credit scoring and fraud detection to bioinformatics and computer vision. In practice, ensemble models such as Random Forests and Gradient Boosted Machines have become standard baselines in machine learning competitions due to their empirical effectiveness.

2.3 Ensemble Learning and Machine Learning Challenges

Despite their advantages, ensemble methods also face challenges that must be addressed to ensure reliable performance. One central issue is the *bias–variance trade-off*. While ensembles are designed to reduce variance through aggregation, they may still inherit the bias of weak base learners or overfit if too many complex models are combined [17]. Striking the right balance between diversity and individual model quality is therefore essential.

Another challenge lies in computational cost. Training multiple models—sometimes hundreds or thousands—can be resource-intensive, particularly in large-scale or real-time applications. Techniques such as parallelization, model pruning, or multi-fidelity optimization have been proposed to reduce the burden while retaining the benefits of ensembles [16].

Interpretability also remains a concern. While single models like decision trees are easily understood, ensembles often function as "black boxes," complicating the explanation of predictions. Recent research has attempted to address this limitation by developing post-hoc explanation methods and by designing ensembles that balance accuracy with interpretability.

Finally, diversity among base learners is a prerequisite for achieving strong ensemble performance. If all learners make similar errors, combining them will not yield substantial improvements [18]. This challenge has motivated the development of diversity measures, hybrid ensemble designs, and advanced combination methods that ensure the ensemble captures complementary information.

In summary, while ensemble learning addresses core challenges of machine learning such as overfitting and instability, it introduces new considerations related to complexity, efficiency, and interpretability. These challenges are at the center of ongoing research in the field.

2.4 Ensemble Learning Categories

Ensemble learning methods can be broadly categorized into three classical families—Bagging, Boosting, and Stacking—along with a range of hybrid and more recent variants. Each approach differs in how base learners are trained and how their predictions are combined, ultimately leading to improvements in generalization, robustness, and stability. The following subsections provide an overview of these categories and highlight their theoretical underpinnings, practical implementations, and applications.

2.4.1 Bagging

Bagging (Bootstrap Aggregating) is one of the earliest and most influential ensemble methods, originally proposed by Breiman [19]. Its primary objective is to reduce variance and improve the stability of predictive models, especially those prone to overfitting such as decision trees. The key idea is to generate multiple bootstrap datasets by randomly sampling the original dataset with replacement. Each bootstrap sample is used to train a separate base learner, ensuring that the models are not identical but instead capture slightly different aspects of the data distribution.

The predictions of the individual models are then aggregated—using majority voting in classification tasks or simple averaging in regression tasks—thereby producing a more stable and accurate final model. This ensemble approach reduces the sensitivity of predictions to small changes in the training data, making it particularly effective for high-variance learners.

A well-known extension of bagging is the Random Forest algorithm, which introduces an additional source of randomness by selecting a random subset of features at each split in the decision trees [11]. This feature-level randomization further enhances diversity among base learners and improves generalization, making Random Forest one of the most widely used algorithms in both academia and industry.

2.4.2 Boosting

Boosting is an ensemble technique that primarily targets the reduction of bias rather than variance [20]. Unlike bagging, which trains models independently and in parallel, boosting builds models sequentially. Each new learner in the sequence focuses on correcting the errors of its predecessors by assigning higher weights to misclassified or poorly predicted samples. This adaptive reweighting strategy forces subsequent learners to concentrate on the most challenging parts of the data.

The aggregated model combines many weak learners, often shallow decision trees (also called decision stumps), into a single strong predictor. The intuition is that while a single weak learner may perform only slightly better than random guessing, the sequential correction process allows the ensemble to achieve high accuracy.

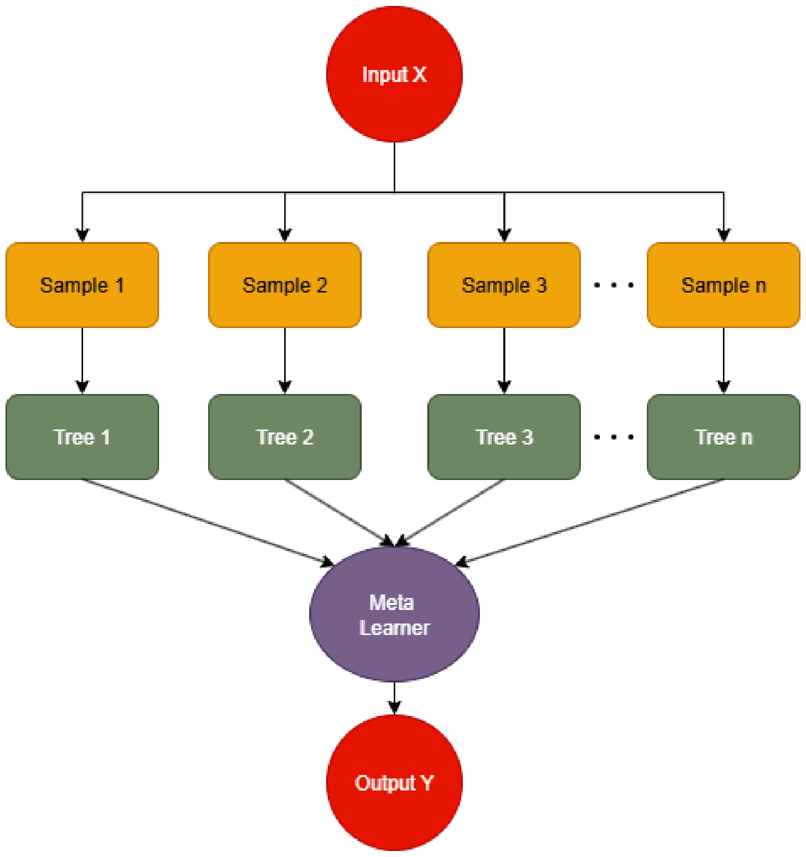


Figure 2.2: Illustration of Bagging: multiple models trained on bootstrap samples are aggregated to produce stable predictions.

Classical boosting algorithms include AdaBoost (Adaptive Boosting), which adaptively reweights samples at each step [20], and Gradient Boosting, which generalizes the idea by optimizing arbitrary loss functions using gradient descent principles [12]. Modern implementations such as XGBoost and LightGBM further improve efficiency through parallelization, regularization, and sophisticated tree-growing strategies, making them state-of-the-art tools in data science competitions and real-world applications.

2.4.3 Stacking

Stacking, also known as stacked generalization, differs fundamentally from bagging and boosting in its aggregation strategy. Instead of relying on uniform averaging or sequential correction, stacking combines the predictions of diverse base learners through a higher-level model called a *meta-learner* [21]. The base learners are typically heterogeneous, including linear models, decision trees, support vector machines, and neural networks. Their predictions on training data are used as input features for the meta-learner, which learns how to optimally combine them to improve predictive performance.

This meta-learning framework exploits the complementary strengths of different algorithms. For example, a linear regression model may capture global linear trends, while a decision tree may capture local non-linear interactions. By combining them, the meta-learner constructs a more accurate and generalizable model.

Stacking has been shown to outperform individual models and even other ensemble methods in many cases, particularly when base learners are sufficiently diverse. It is widely applied in complex real-world tasks such as bioinformatics, text classification, and financial forecasting, where robustness and interpretability are essential.

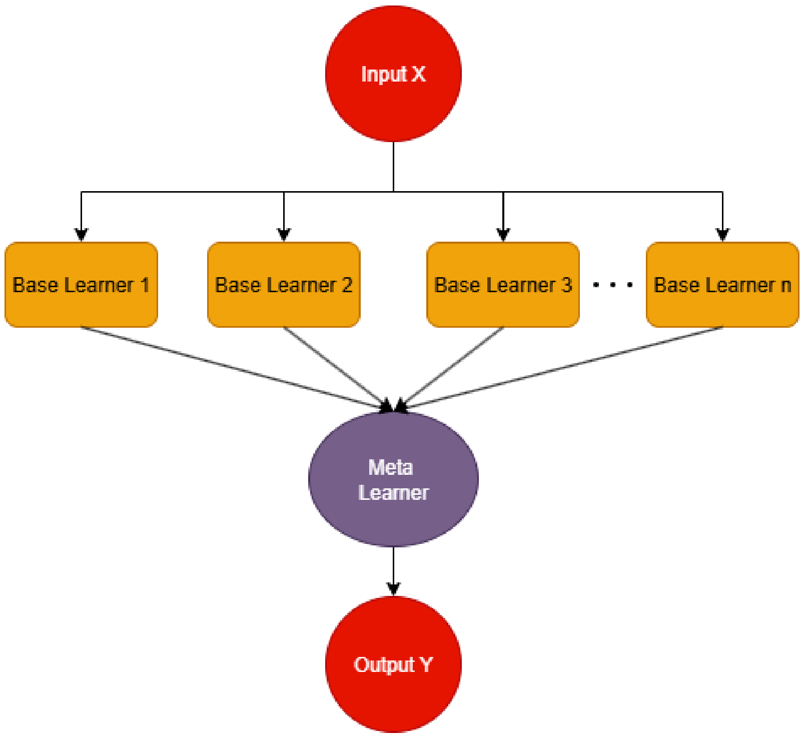


Figure 2.3: Illustration of Stacking: predictions of heterogeneous base learners are combined through a meta-learner.

2.4.4 Hybrid and Recent Variants

Hybrid ensembles combine ideas from bagging, boosting, and stacking, often tailoring them to specific problem domains. For instance, researchers have proposed applying bagging to boosting models to stabilize their variance, or using boosting within stacking frameworks to enhance predictive accuracy. Recent developments include *dynamic ensemble selection*, where only the most competent learners are chosen based on the characteristics of each test instance, and *ensemble deep learning*, which integrates classical ensemble strategies with deep neural networks for tasks such as image recognition and natural language processing [18].

These hybrid and emerging approaches highlight the flexibility of ensemble learning. By leveraging the complementary benefits of multiple strategies, they continue to push the boundaries of predictive performance, offering reliable solutions for increasingly complex and large-scale machine learning problems.

2.5 Combination Methods

The predictive power of an ensemble model is determined not only by the choice of base learners but also by the strategy used to combine their outputs into a single decision function. Combination methods aim to leverage the diversity of base learners while minimizing their individual weaknesses, such as overfitting or limited capacity. This section presents three widely used combination techniques: simple averaging, weighted averaging, and meta-learning (stacking). Each method differs in complexity, flexibility, and computational cost, but all share the goal of improving generalization performance by exploiting the strengths of multiple learners [18].

2.5.1 Simple Averaging

Simple averaging is the most straightforward combination method and is often used as a baseline in ensemble learning research. The idea is to compute the arithmetic mean of predictions from all base learners, without assigning them different levels of importance. Formally, given M base learners $h_1, h_2, \ldots, h_M$, the final prediction for an instance x_i is defined as:

$$H(x_i) = \frac{1}{M} \sum_{j=1}^{M} h_j(x_i), \tag{2.12}$$

where $h_j(x_i)$ denotes the output of the j-th learner for the input x_i [6].

This approach is particularly effective when base learners are unstable (e.g., decision trees) and exhibit high variance. By averaging, fluctuations cancel each other out, leading to a more stable and generalizable prediction. However, simple averaging implicitly assumes that all learners contribute equally, which may not be ideal when some models are clearly more accurate than others [2]. Despite this limitation, its computational simplicity makes it highly attractive, especially in large-scale applications or real-time systems where interpretability and speed are critical.

2.5.2 Weighted Averaging

Weighted averaging generalizes the idea of simple averaging by assigning weights to the predictions of each learner. In this way, stronger learners are given more influence, while weaker learners contribute less. The prediction is formally written as:

$$H(x_i) = \sum_{j=1}^{M} w_j h_j(x_i), \tag{2.13}$$

where the weights w_j satisfy the following constraints:

$$w_j \geq 0 \quad \text{and} \quad \sum_{j=1}^{M} w_j = 1. \tag{2.14}$$

The determination of weights w_j can follow various strategies. One common approach is to optimize them on a validation set using performance metrics such as mean squared error for regression or cross-entropy for classification. Alternatively, gradient-based optimization methods can be employed to fine-tune the weights jointly with other model parameters [17]. In heuristic approaches, weights may be determined by rules such as proportionality to base learner accuracy or inverse error rates.

Weighted averaging offers greater flexibility than simple averaging because it accounts for differences in learner performance. For instance, in a heterogeneous ensemble that combines linear regression, decision trees, and neural networks, assigning appropriate weights prevents weaker models from degrading the ensemble's accuracy. However, weight optimization introduces an additional layer of complexity and may itself lead to overfitting if not regularized properly. Despite this, weighted averaging is widely applied in ensemble learning and often serves as a strong baseline in competitions and applied research.

2.5.3 Meta-Learning (Stacking)

Meta-learning, or stacked generalization, represents a more advanced combination method in which the predictions of base learners are not aggregated by a fixed rule, but instead fed into a second-level learning algorithm called a *meta-learner*. The meta-learner is trained to learn the optimal way of combining the outputs of the base models. Formally, given predictions $h_1(x), h_2(x), \ldots, h_M(x)$ from M base learners, the final prediction is defined as:

$$H(x) = T(h_1(x), h_2(x), \ldots, h_M(x)), \tag{2.15}$$

where $T(\cdot)$ denotes the meta-learner, which can itself be any supervised learning model [21].

The choice of meta-learner significantly influences the success of stacking. Linear models such as logistic regression are often used due to their simplicity and interpretability, while more complex learners like random forests or gradient boosting machines can capture non-linear dependencies between base model predictions. Stacking excels in scenarios where base learners capture complementary aspects of the data. For example, a support vector machine might model local decision boundaries effectively, while a neural network captures complex feature interactions. The meta-learner combines these strengths, often achieving superior accuracy compared to bagging or boosting [18].

However, stacking also introduces additional challenges. It requires careful cross-validation to avoid overfitting, since the meta-learner is trained on the predictions of base learners, which themselves are fitted on the training data. Moreover, stacking tends to be computationally expensive because it requires multiple layers of training. Despite these challenges, stacking remains a powerful and widely used ensemble strategy, particularly in high-stakes applications and machine learning competitions.

2.6 Diversity in Ensemble Learning

One of the fundamental principles underlying the success of ensemble learning is the concept of *diversity* among base learners. While individual models may perform reasonably well, their errors are often correlated. If the ensemble consists of highly similar models, then combining them yields limited improvement. In contrast, when learners are diverse—making different types of errors—their combination can substantially improve generalization by averaging out individual weaknesses [22]. Therefore, diversity is a key factor determining the effectiveness of ensemble methods.

2.6.1 Importance of Diversity

Diversity is particularly important in reducing the risk of overfitting and improving robustness to noisy or imbalanced datasets. An ensemble of identical learners would perform no better than any of its components, but introducing diversity allows the ensemble to exploit complementary information among models [17]. This leads to lower variance and, in some cases, even reduced bias, depending on the underlying learning strategy.

Furthermore, diversity enables ensembles to handle heterogeneous data sources and complex real-world tasks. For example, in medical diagnosis or financial forecasting, different learners may capture distinct relationships, and their combination can provide more reliable predictions than a single model. However, it is crucial to balance diversity with accuracy: highly diverse but inaccurate learners will degrade performance, while highly accurate but identical learners offer no ensemble benefit [23]. Thus, the challenge lies in constructing ensembles where individual models are both accurate and diverse.

2.6.2 Classical Diversity Measures (Q-Statistic, Correlation, Disagreement)

Quantifying diversity in ensemble learning has long been recognized as a crucial step towards understanding and improving ensemble performance. To this end, several classical diversity measures have been introduced, among which the most widely used are the Q-statistic, correlation, and disagreement measures [22, 23].

Q-Statistic
The Q-statistic, first proposed in the context of ensemble methods by [24], measures the degree of association between the predictions of two classifiers. Given two classifiers C_i and C_j, the Q-statistic is defined as:

$$Q_{ij} = \frac{N_{11}N_{00} - N_{10}N_{01}}{N_{11}N_{00} + N_{10}N_{01}}, \tag{2.16}$$

where N_{11} is the number of samples both classifiers predict correctly, N_{00} is the number both predict incorrectly, and N_{10}, N_{01} are the counts of disagreements. The value of Q_{ij} ranges from -1 (complete disagreement) to $+1$ (complete agreement), with 0 indicating independence. Lower correlations (values closer to zero or negative) indicate higher diversity.

Correlation Coefficient
The correlation coefficient, adapted from classical statistics, measures the linear relationship between the outputs of classifiers. For two classifiers C_i and C_j, the correlation is given by:

$$\rho_{ij} = \frac{\text{cov}(C_i, C_j)}{\sqrt{\text{var}(C_i) \cdot \text{var}(C_j)}}, \tag{2.17}$$

where cov denotes covariance and var variance. High positive correlation implies redundancy in predictions, while low or negative correlation indicates diversity. However, this measure is sensitive to class imbalance and may overestimate diversity in skewed datasets [22].

Disagreement Measure
The disagreement measure directly quantifies the proportion of instances on which two classifiers disagree. It is defined as:

$$D_{ij} = \frac{N_{10} + N_{01}}{N}, \tag{2.18}$$

where N is the total number of samples. Unlike correlation-based metrics, the disagreement measure is simple, interpretable, and particularly effective in binary classification problems. A higher disagreement score indicates greater diversity among classifiers [17].

Although each of these classical metrics provides valuable insights, they are often insufficient on their own, as they do not capture the complex interplay between diversity and accuracy in multi-class or regression settings. This limitation has led to the development of more advanced approaches, as discussed in the next subsection.

2.6.3 Recent Approaches

While classical diversity measures such as Q-statistic, correlation, and disagreement remain widely used, recent research has highlighted their limitations in capturing the nuanced relationship between diversity and ensemble accuracy. Consequently, more sophisticated measures and frameworks have been proposed in the literature [25, 26, 27].

One notable development is the use of **information-theoretic measures**, such as entropy and mutual information, to quantify the uncertainty and complementarity among base learners [25]. These approaches extend beyond pairwise comparisons, capturing the joint diversity of the ensemble as a whole.

Another line of research has introduced **margin-based measures**, which evaluate diversity by analyzing the distribution of classification margins— that is, the difference between the correct class score and the maximum incorrect class score. Ensembles with larger margins tend to be both more accurate and more robust, motivating metrics such as the margin distribution variance [28].

In addition, **dynamic diversity measures** have been developed to account for test-time variability. Instead of computing static diversity values on training data, these methods adaptively estimate diversity for each test instance, allowing for dynamic ensemble selection strategies [29]. Such approaches are particularly valuable in non-stationary environments, where data distributions evolve over time.

Finally, recent studies have emphasized the interplay between **diversity and interpretabi-**

lity. Hybrid approaches that combine classical diversity measures with model explanation techniques (e.g., SHAP or LIME) aim to build ensembles that are not only accurate but also transparent [18]. This reflects the growing importance of explainable AI in critical domains such as healthcare and finance.

In summary, while classical measures provide foundational tools for analyzing diversity, modern approaches emphasize richer statistical, dynamic, and interpretability-driven perspectives. Together, they contribute to a more comprehensive understanding of how diversity influences ensemble learning.

2.7 Scoring Metrics in Ensemble Learning

The evaluation of ensemble models requires suitable scoring metrics that capture not only predictive accuracy but also calibration, robustness, and generalization ability [6]. Since ensembles often aggregate heterogeneous learners, metrics must reflect both discrimination (e.g., accuracy, precision, recall, AUC) and reliability (e.g., log-loss, Brier score). In regression tasks, common measures include RMSE, MAE, and R^2 [30]. Appropriate scoring metrics thus provide a principled basis for comparing ensembles and guiding their optimization.

2.7.1 Why We Need Scoring Metrics

Scoring metrics are essential because they define what "good performance" means for a given task. They provide objective criteria for model selection, hyperparameter tuning, and comparison across datasets [31]. Without proper metrics, ensembles may appear effective but fail to generalize. Moreover, metrics highlight task-specific trade-offs, such as precision vs. recall in imbalanced data, or error magnitude vs. variance in regression. In this sense, metrics act as a bridge between statistical performance and real-world application goals.

2.7.2 Covariance

Covariance measures the joint variability between two sets of predictions, providing insight into the linear association and, in the context of ensemble learning, the diversity between two base learners' outputs. A low covariance implies high diversity, while a high covariance indicates similar predictive behaviors [22, 23]. Given two vectors $\mathbf{a}$ and $\mathbf{b}$:

$$\mathrm{Cov}(\mathbf{a}, \mathbf{b}) = \frac{1}{n-1} \sum_{i=1}^{n} (a_i - \bar{a})(b_i - \bar{b})$$

where $\bar{a}$ and $\bar{b}$ are the means.

2.7.3 Pearson Correlation

Pearson correlation coefficient quantifies the strength and direction of the linear relationship between two variables. In ensemble learning, it is commonly used to evaluate the similarity between the predictions of different base learners. Lower correlation suggests greater diversity [32, 33]. It is calculated as:

$$r_{ab} = \frac{\sum_{i=1}^{n}(a_i - \bar{a})(b_i - \bar{b})}{\sqrt{\sum_{i=1}^{n}(a_i - \bar{a})^2}\sqrt{\sum_{i=1}^{n}(b_i - \bar{b})^2}}$$

where a_i and b_i are predictions and $\bar{a}, \bar{b}$ are means.

2.7.4 Spearman's Rank Correlation

Spearman's rank correlation assesses the monotonic relationship between two variables based on their ranks, not their absolute values. It is robust to outliers and captures non-linear but monotonic relationships. In the context of ensemble diversity, it captures agreement in the order of predictions [34, 35]. The formula is:

$$r_{s,ab} = 1 - \frac{6\sum_{i=1}^{n} d_i^2}{n(n^2 - 1)}$$

where d_i is the difference in ranks of a_i and b_i.

2.8 Advanced Diversity Metrics in Ensemble Regression

While traditional diversity measures such as correlation, Q-statistic, and disagreement provide useful insights, they remain limited in their ability to explain the interplay between diversity and accuracy in ensemble regression settings. Recent work has highlighted that these metrics often fail to capture whether diversity necessarily leads to better predictive performance or how error structures among base learners can complement each other [36, 37]. To address these limitations, novel approaches have been introduced, most notably the *Sign Diversity Coefficient (SDC)* and the *Extended Weighted Sign Disagreement Metric (EWSDM)* [38, 39]. These methods explicitly integrate accuracy and diversity into a unified framework and are particularly well suited for ensemble regression tasks.

2.8.1 Sign Diversity Coefficient (SDC)

The Sign Diversity Coefficient (SDC) is motivated by the observation that base learners (BLs) may systematically *overestimate* or *underestimate* target values. If all BLs share the same bias direction, their aggregation risks amplifying error in one direction. Conversely, if some BLs overpredict while others underpredict, their errors can cancel out, resulting in a more balanced ensemble prediction [39].

For each data point i, we define:

$$O_i = \sum_{m=1}^{M} I(\hat{y}_{mi} > y_i), \qquad U_i = \sum_{m=1}^{M} I(\hat{y}_{mi} < y_i), \tag{2.27}$$

where O_i counts the number of BLs that overestimate and U_i counts those that underestimate the true target y_i. The sign diversity for that data point is then measured as:

$$SDC_i = \min(O_i, U_i). \tag{2.28}$$

The overall SDC for N samples is:

$$SDC = \sum_{i=1}^{N} SDC_i. \tag{2.29}$$

This measure ranges from 0 (no diversity in sign; all BLs predict in the same direction) to higher values that reflect disagreement across BLs. For instance, in an ensemble of three learners, if two overestimate and one underestimates, the SDC_i equals 1. In larger ensembles, the measure captures the minority-side predictions, emphasizing how many BLs disagree with the majority at each data point [38].

Illustrative Example. Consider four BLs predicting the same data point. If all overestimate, $O = 4$, $U = 0$, and $SDC = 0$. If three overestimate and one underestimates, $O = 3$, $U = 1$, then $SDC = 1$. If predictions are evenly split ($O = 2$, $U = 2$), then $SDC = 2$, indicating maximum diversity for this case. This makes SDC particularly interpretable and useful for analyzing how prediction directions complement one another.

Significance. Unlike correlation-based metrics, SDC directly captures structural complementarity in prediction errors. This is critical in regression, where ensembles aim not only to reduce variance but also to mitigate systematic bias [40]. By quantifying sign disagreement, SDC highlights ensembles where base learners "cover" each other's weaknesses, producing more robust aggregated predictions.

2.8.2 Extended Weighted Sign Disagreement Metric (EWSDM)

Building on SDC, the Extended Weighted Sign Disagreement Metric (EWSDM) provides a unified framework that balances predictive accuracy and diversity. It extends the earlier WSDM (defined for pairs of BLs) to ensembles of arbitrary size [38].

The EWSDM is defined as:

$$EWSDM = \alpha \cdot \left(\frac{1}{M} \sum_{i=1}^{M} R_i^2 \right) + (1 - \alpha) \cdot \frac{SDC}{N}, \tag{2.30}$$

where:

- R_i^2 is the coefficient of determination of base learner i, reflecting predictive accuracy;
- SDC is the overall sign diversity (Equation 2.29);
- $\alpha \in [0, 1]$ is a trade-off parameter balancing accuracy and diversity.

Interpretation. The first component measures the average explanatory power of the BLs. The second captures sign-based disagreement across the ensemble. By placing both terms on the same $[0, 1]$ scale, EWSDM ensures comparability, allowing α to control the balance explicitly. A higher α favors accuracy, whereas a lower α emphasizes diversity.

Advantages. EWSDM addresses two major limitations of classical metrics: (1) they evaluate diversity independently of accuracy, and (2) they are often restricted to pairwise comparisons [36, 41]. By integrating both dimensions, EWSDM produces a single interpretable score that can guide ensemble construction. It supports base learner selection, weighting strategies, and optimal ensemble sizing. Studies have shown that ensembles optimized with EWSDM achieve higher robustness and generalization in regression tasks compared to those selected by correlation or Q-statistic alone [39, 42].

Practical Use. In practice, EWSDM can be visualized using contour plots, where the x-axis represents normalized SDC values and the y-axis average R^2. Higher EWSDM scores appear in the top-right region, corresponding to ensembles that achieve both strong predictive performance and diverse error structures [38]. This provides a practical tool for evaluating candidate ensembles in stacking frameworks.

2.8.3 Implications for Ensemble Learning

The introduction of SDC and EWSDM marks a shift towards metrics that are *task-aware* and *bias-sensitive*. They reflect not only variance reduction but also how error directions interact among learners. This perspective aligns with recent trends in ensemble research, emphasizing the synergy of base learners rather than their isolated performance [43, 36]. Moreover, by embedding both accuracy and diversity into a single criterion, EWSDM simplifies the model selection process and reduces the risk of overfitting to spurious diversity signals.

In summary, SDC provides a clear lens into the structural disagreement of predictions, while EWSDM extends this idea into a flexible scoring system for ensembles of any size. Together, they advance ensemble methodology by offering interpretable, scalable, and accuracy-conscious diversity measures, setting a foundation for more balanced and generalizable ensemble regression models.

3 Automated Optimization of Ensemble Size and the Diversity–Accuracy Trade-Off in Sign Diversity Metric

3.1 Existing Framework

The baseline framework for ensemble regression aims to construct stacking ensembles by leveraging the complementary strengths of a heterogeneous pool of base learners (BLs). The pipeline begins with data handling and validation. To mitigate variance in performance estimates, the dataset is repeatedly partitioned into training and testing sets using a repeated hold-out or k-fold cross-validation scheme. Each split is initialized with a different random seed, ensuring that the evaluation does not depend on a single data partition. While this procedure provides robustness for medium-sized datasets, it also reduces the effective training size and produces unstable estimates in small, high-dimensional datasets, which represent the most challenging and practically relevant regression problems.

The candidate pool of BLs encompasses a wide range of algorithms, including linear models such as Linear Regression and Lasso, kernel-based approaches like Kernel Ridge Regression and Support Vector Regression, tree-based methods such as Decision Trees, Random Forests, and Gradient Boosting, as well as non-parametric and instance-based models including k-Nearest Neighbors, Gaussian Process Regression, and Multi-layer Perceptrons. Each of these algorithms is defined over a structured hyperparameter search space, allowing systematic exploration of model configurations. By assembling such a diverse pool, the framework seeks to maximize heterogeneity in inductive biases and error patterns, thus providing fertile ground for ensemble diversity.

Once the pool of candidate learners has been trained, ensembles of size $m \in \{2, 3, 4\}$ are formed by combining different subsets of models. The quality of each ensemble is evaluated according to the Extended Weighted Sign Disagreement Metric (EWSDM), which explicitly balances predictive accuracy and diversity. Accuracy is measured via the coefficient of determination (R^2), while diversity is quantified through the Sign Diversity Coefficient (SDC), capturing the extent to which learners overestimate or underestimate the target in opposing directions. The trade-off between these two components is governed by a parameter $\alpha \in [0, 1]$: higher values emphasize accuracy, whereas lower values prioritize diversity.

In the existing implementation, however, the parameter α is not optimized adaptively but instead tuned through a simple grid search over a small set of fixed values, typically $\alpha \in \{0.25, 0.5, 0.75\}$. For each ensemble size and each preset α, the framework exhaustively evaluates all possible model combinations. While this approach provides a straightforward way to explore the interaction between accuracy and diversity, it remains limited in resolution and computational efficiency. Potentially optimal trade-offs lying between the fixed grid values may remain undiscovered, and resources are wasted on evaluating combinations that are unlikely to be competitive. Moreover, ensemble size and α are treated as independent factors, rather than being optimized jointly, which prevents the framework from fully capturing their interaction in determining ensemble performance.

Stacking is employed to aggregate the selected base learners into a single ensemble predictor. The individual predictions of BLs are treated as meta-features and passed to a meta-learner, most often Kernel Ridge Regression (KRR), which has shown strong empirical performance in balancing bias and variance. Finally, the ensembles are assessed not only through EWSDM scores but also by applying a bias–variance–diversity decomposition, which provides further insight into how the chosen BLs contribute to generalization.

At the conclusion of this procedure, the framework outputs the ensemble size, the set of selected BLs with their tuned hyperparameters, the α value associated with the best trade-off, and the resulting evaluation metrics. This design successfully demonstrates the potential of sign diversity as a meaningful criterion for ensemble construction, yet it remains computationally demanding, limited by the coarse granularity of grid search, and suboptimal for small datasets where k-fold validation provides unstable estimates. These shortcomings motivate the need for a more efficient and automated optimization approach that integrates validation, diversity–accuracy balancing, and ensemble size selection within a unified framework.

3.2 Problem Statement

Although the existing framework demonstrates the potential of sign diversity as a guiding principle for ensemble regression, several methodological and computational limitations remain unresolved, restricting its effectiveness in practice. A key concern is the reliance on k-fold cross-validation for performance estimation. While k-fold CV is a standard approach in machine learning, its application to small, high-dimensional datasets is problematic. Dividing the data into folds reduces the already limited training sample size, thereby inflating variance in model evaluation and producing unstable estimates of generalization performance. For domains where data are scarce but complex, such as material sciences or industrial quality prediction, these instabilities render k-fold validation inadequate and call for more robust alternatives.

Another critical limitation lies in the tuning of the diversity–accuracy trade-off parameter α and the ensemble size. In the current framework, both are selected through a sequential grid search, in which ensembles of size $m \in \{2, 3, 4\}$ are tested across a fixed grid of α values, typically $\{0.25, 0.5, 0.75\}$. While this approach ensures coverage of a few representative configurations, it remains inherently restricted. Grid search is computationally expensive, as each configuration must be evaluated independently, and its coarse granularity risks missing the true optimum. Potentially superior ensembles may exist at intermediate α values or at ensemble sizes outside the restricted set, but these possibilities are ignored. Furthermore, treating α and ensemble size as independent search dimensions prevents systematic exploration of their interaction, even though their combined influence strongly determines ensemble performance.

These methodological issues are compounded by computational inefficiency. The evaluation loop requires training and validating a large number of base learners and their combinations, each assessed under repeated cross-validation. As a result, runtime grows rapidly with the number of learners and candidate hyperparameters. This inefficiency is particularly problematic when more robust validation techniques, such as leave-one-out cross-validation (LOOCV), are considered. LOOCV is statistically more appropriate for small datasets, as it maximizes the use of available data and provides nearly unbiased estimates of generalization error, but it also multiplies the number of evaluations required. Without additional strategies to accelerate computations, such as parallelization, the framework becomes impractically slow.

In summary, the current approach to ensemble regression, while conceptually sound, is limited by three factors: (i) unreliable performance estimation for small datasets due to k-fold CV, (ii) inefficient and coarse hyperparameter tuning of α and ensemble size through grid search, and (iii) prohibitive computational cost when scaling to more robust validation or larger search spaces. These shortcomings motivate the development of a more efficient and automated optimization pipeline in which LOOCV provides statistically reliable validation, parallelization alleviates the associated computational burden, and Bayesian optimization replaces grid search to adaptively tune ensemble size and α as joint hyperparameters. Such a framework promises to construct ensembles that are both more accurate and more diverse, while remaining computationally feasible for small, high-dimensional regression problems.

3.3 Leave-One-Out Cross-Validation (LOOCV) and Parallel Computing

3.3.1 Leave-One-Out Cross-Validation: General Overview

Cross-validation (CV) is a fundamental technique for assessing how well the predictions of machine learning models generalize to an independent data set. Among various CV techniques, Leave-One-Out Cross-Validation (LOOCV) is particularly powerful when dealing with datasets characterized by limited sample size. LOOCV systematically leaves out one sample for testing and utilizes the remaining $n-1$ samples for training. This process repeats n times—once for each data sample—ensuring exhaustive and unbiased performance evaluation.

In LOOCV, each observation is used exactly once as a test sample and $n-1$ times for model training. Consequently, LOOCV provides nearly unbiased estimates of model generalization capability. Furthermore, LOOCV is particularly valuable in scenarios involving limited data points, as it maximizes the use of available data. This methodology also allows for precise analysis of individual errors and facilitates the detection of influential or anomalous data points.

Figure 3.1 clearly illustrates the LOOCV process. Each model iteration is visualized separately, with one sample excluded from the training set (marked in blue) and all other samples included (marked in orange). Thus, this process ensures that every sample in the dataset has a corresponding prediction generated by a model that was trained independently from that specific sample.

However, despite its methodological advantages, LOOCV involves substantial computational demands. Specifically, the necessity to train the model n separate times dramatically increases runtime, especially with computationally intensive algorithms or extensive

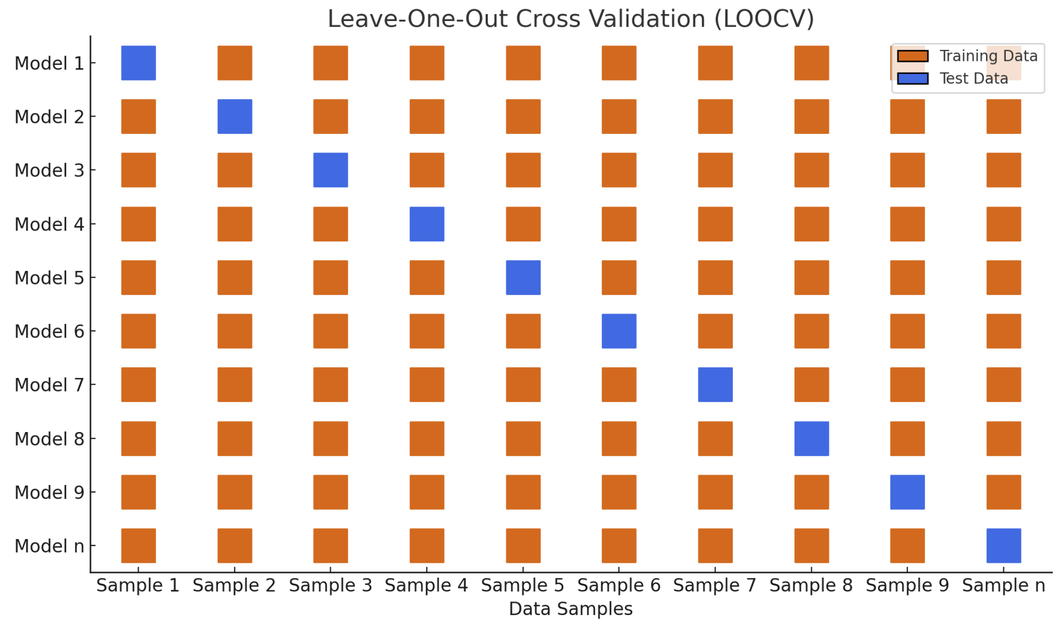


Figure 3.1: Schematic representation of the Leave-One-Out Cross-Validation process. Each blue square represents the single test data point, while orange squares represent training data points.

hyperparameter optimization tasks.

3.3.2 Integration into the Existing Framework

The integration of LOOCV into the existing ensemble learning framework requires substantial modification of the validation pipeline. In the original design, model performance was estimated using repeated hold-out or k-fold cross-validation, where datasets were partitioned into subsets and average performance was reported across repetitions. While effective for medium-sized datasets, this approach proved less reliable for small, high-dimensional data, where partitioning further reduces the available training samples and leads to unstable generalization estimates. Replacing this validation procedure with LOOCV ensures that every data point is used as an independent test instance, thereby maximizing statistical efficiency and delivering nearly unbiased estimates of ensemble performance.

Within the revised workflow, LOOCV is implemented at multiple stages. First, each base learner is evaluated under LOOCV to ensure that predictive quality is measured on the full dataset without excluding potentially influential samples. Second, ensembles of different sizes and trade-off parameters are constructed, and their Extended Weighted Sign Disagreement Metric (EWSDM) values are computed with LOOCV-based estimates of R^2. This guarantees that both accuracy and diversity are assessed under the same rigorous validation regime. Finally, during hyperparameter optimization, LOOCV serves as the objective evaluation step for Bayesian optimization, providing the optimizer with high-fidelity feedback about the true generalization behavior of each candidate configuration.

Implementing LOOCV in this manner required structural changes to the training loop. Instead of a fixed number of k partitions, the pipeline now generates n iterations, each leaving one observation aside for testing. The corresponding predictions are stored for later aggregation into ensemble metrics, and the overall process yields a comprehensive mapping between ensemble configurations and their LOOCV performance. This design ensures that the framework fully exploits limited data while maintaining compatibility with the EWSDM objective, the stacking procedure, and the hyperparameter optimization module.

3.3.3 Computational Challenges and the Necessity of Parallelization

For this research, involving a dataset of 139 observations, LOOCV requires training the model 139 times per evaluation. Consequently, this process leads to a significant computational burden, which often becomes impractical or excessively time-consuming in practice, particularly when utilizing complex model structures or iterative optimization approaches like Bayesian optimization.

To address these challenges, parallel computing strategies are employed, effectively distributing computational tasks across multiple CPUs or cores. Parallelization not only reduces the total runtime but also enhances computational efficiency and ensures feasibility for complex modeling and extensive evaluation processes.

3.3.4 Parallel Computing

Parallel computing refers to the simultaneous execution of multiple computational tasks, designed to accelerate workloads that would otherwise be processed sequentially. Its primary goal is to exploit the availability of modern multi-core processors or distributed computing systems, thereby reducing runtime and improving efficiency. By dividing large problems into smaller independent units and processing them concurrently, parallel computing enables significant gains in performance, particularly in applications where repeated model training and evaluation are required.

In the context of machine learning, parallel computing is especially valuable for tasks involving cross-validation or hyperparameter optimization, both of which require multiple independent model evaluations. Since each evaluation can be performed without depending on the outcome of others, these tasks are inherently parallelizable. Distributing them across multiple processing cores allows researchers to explore larger search spaces, adopt more robust validation schemes, and achieve results within reasonable computational budgets.

Figure 3.2 illustrates the general concept of parallel computing in the Leave-One-Out Cross-Validation (LOOCV) workflow. The dataset is divided into folds, each fold is assigned to a separate CPU core, and model training and testing for that fold are executed independently. After all folds have been processed, their results are aggregated to produce the final performance estimate. This process not only shortens computation time but also ensures scalability when working with increasingly complex ensemble configurations.

The main advantages of parallel computing lie in its ability to reduce training time, leverage existing hardware resources more effectively, and make otherwise computationally demanding procedures feasible. Within ensemble learning, where multiple base learners and hyperparameter configurations must be tested, parallel computing thus serves as an indispensable tool, enabling more comprehensive and efficient experimentation.

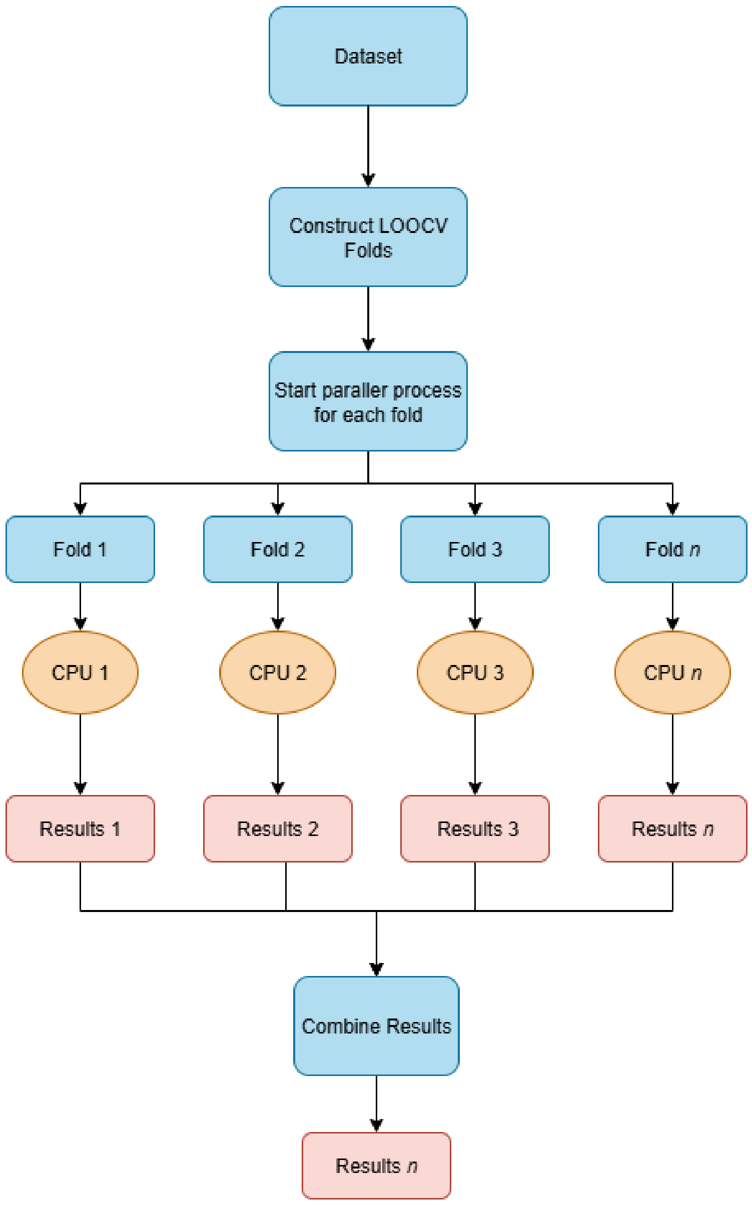


Figure 3.2: Parallel LOOCV workflow. Independent folds are assigned to separate CPU cores, processed concurrently, and their results combined after completion.

3.3.5 Parallel Computing Implementation

To render LOOCV tractable under extensive model and hyperparameter searches, the validation loop is parallelized at the *fold level*. The implementation adopts a process-based, embarrassingly parallel scheme: each leave-one-out fold (exactly one test instance and $n-1$ training instances) is evaluated in an independent worker process, and fold-wise results are consolidated afterwards. This strategy aligns naturally with LOOCV's independence assumptions across folds and avoids synchronization hotspots during model fitting.

Concretely, fold-level parallelism is realized via `joblib.Parallel` with the `loky` backend (default), which spawns separate Python processes to bypass the GIL and achieves true CPU parallelism. The driver routine constructs a list of fold indices $i \in \{0, \ldots, n-1\}$

and dispatches one task per fold:

$$\texttt{fold\_dicts} \leftarrow \texttt{Parallel(n\_jobs = }C\texttt{)}\big(\texttt{delayed(run\_loocv\_fold)}(i,\ldots)\big)_{i=0}^{n-1},$$

where C denotes the number of available CPU cores. Each worker executes the full training–validation pipeline for its assigned fold: (i) define train/test split by excluding index i; (ii) scale inputs with a `MinMaxScaler`; (iii) enumerate base-learner combinations of target sizes; (iv) for each combination, perform inner Bayesian hyperparameter search on the training subset; (v) refit with the best configuration and evaluate on the held-out instance; (vi) compute ensemble diagnostics (e.g., bias–variance–diversity) and stacking outcomes. The worker returns a compact dictionary keyed by combination identifiers and downstream metrics (e.g., stacking MSE), which are aggregated by the parent process via a sequential reduce step.

Resource isolation and avoidance of nested parallelism. To prevent oversubscription and contention between process-level parallelism and BLAS/OpenMP threading, the implementation explicitly pins numerical libraries to a single thread per worker by setting configuration `OMP_NUM_THREADS=1`, `MKL_NUM_THREADS=1`, `OPENBLAS_NUM_THREADS=1`, `VECLIB_MAXIMUM_THREADS=1`, and `NUMEXPR_NUM_THREADS=1` prior to spawning workers. Additionally, inner estimators and search routines are configured with `n_jobs=1` (e.g., `BayesSearchCV` for the meta-learner), thereby disabling nested parallel regions and ensuring stable wall-clock scaling with the number of outer processes. This "one thread per worker" policy eliminates resource contention and yields near-linear speedups up to the CPU core count, subject to I/O and memory constraints.

Determinism and fault tolerance. Each worker process is seeded deterministically (fold index i is reused as `random_state`) for reproducibility of stochastic training procedures and Bayesian search initialization. Transient numerical and convergence warnings are suppressed locally to keep workers independent and robust. When an inner model fit fails for a proposed hyperparameter vector, the objective function returns a large sentinel loss (e.g., 10^6), allowing the Bayesian optimizer to continue without crashing the worker; this yields graceful degradation rather than job aborts.

Interface with Bayesian optimization for α. Parallel LOOCV is embedded inside the outer hyperparameter routine that tunes the diversity–accuracy trade-off $\alpha \in [0, 1]$ by Gaussian-process–based global optimization (`skopt.gp_minimize`). For a proposed α, the objective function launches the fold-parallel evaluation described above and aggregates per-fold metrics into an objective value (e.g., average test MSE over the best combinations at each target ensemble size). Thus, the optimizer receives high-fidelity, LOOCV-based feedback for each α candidate while the parallel engine amortizes the associated computational cost.

I/O discipline and result aggregation. To avoid race conditions during concurrent writes, each worker emits its diagnostics into per-fold, per-iteration files under unique names (or returns data in-memory), and a subsequent sequential aggregation step computes across-fold summaries (means, variances) and selects the incumbent ensemble configuration. This discipline guarantees output integrity without inter-process locks or shared-state synchronization.

Scalability considerations. The maximum degree of parallelism is bounded by $\min(n, C)$, where n is the sample size (number of LOOCV folds) and C is the number of hardware cores. Memory footprint scales with the number of resident workers because each process holds its own model instances and data views; hence `n_jobs` is set conservatively with respect to RAM. Since inner model fits are CPU-bound and largely data-local, load balancing is effective even with heterogeneous base learners and search spaces. In practice, this design reduces LOOCV wall-clock from $\mathcal{O}(n)$ serial time to approximately $\mathcal{O}(n/C)$, enabling the use of exhaustive LOOCV within the ensemble-selection pipeline.

Software components. The implementation builds upon `joblib` for process-based parallelism and function serialization, `scikit-learn` for estimators, preprocessing, and stacking, and `scikit-optimize` for Bayesian hyperparameter search. These components provide a stable, battle-tested substrate for parallel model selection and evaluation in Python, facilitating concise orchestration code while preserving reproducibility and portability.

3.4 Bayesian Optimization of the Alpha Parameter

3.4.1 Significance of the Alpha Parameter in the EWSDM Metric

The hyperparameter α in the EWSDM metric plays a critical role in explicitly balancing predictive accuracy and diversity in ensemble regression. As defined in Equation 3.1,

$$EWSDM = \alpha \left(\frac{1}{M} \sum_{i=1}^{M} R_i^2 \right) + (1-\alpha) \frac{SDC}{N}, \tag{3.1}$$

the contribution of each term is directly controlled by α. When α approaches 1, the EWSDM reduces to a formulation dominated by the accuracy term:

$$\lim_{\alpha \to 1} EWSDM = \frac{1}{M} \sum_{i=1}^{M} R_i^2,$$

which corresponds to selecting ensembles almost exclusively on the basis of their average predictive performance. In this case, diversity contributes negligibly, and the framework risks producing ensembles composed of highly accurate but correlated learners.

Conversely, when α approaches 0, the EWSDM emphasizes only the diversity component:

$$\lim_{\alpha \to 0} EWSDM = \frac{SDC}{N},$$

favoring ensembles where base learners disagree substantially in their prediction signs. Although this can maximize error complementarity, it may also incorporate weak learners with poor accuracy, potentially harming generalization if diversity is prioritized excessively.

Thus, the α parameter provides a continuous mechanism to navigate the bias–variance–diversity trade-off. Intermediate values of α yield a balance between the explanatory power of base learners and their disagreement structure, ensuring that the ensemble exploits both predictive accuracy and error complementarity.

Importantly, the optimal α is highly dataset-dependent. For datasets with strong, stable predictors, larger values of α can be advantageous as they emphasize accuracy. In contrast, for complex or noisy domains where individual learners tend to overfit or produce correlated errors, smaller values of α help to exploit diversity and improve robustness.

In summary, α is not a trivial weighting factor but the key control knob of the EWSDM metric. By adjusting α, practitioners can adapt the ensemble evaluation criterion to the specific statistical properties of their data, thereby directly influencing the generalization ability of the resulting ensemble model.

3.4.2 Implementation Details

The Bayesian optimization of the α parameter was implemented using the `gp_minimize` function provided by the `scikit-optimize` library [44]. This function applies Gaussian Process regression to approximate the unknown objective function, while an acquisition function guides the exploration of the parameter space by balancing exploitation of promising regions with exploration of areas of high uncertainty. The design of the pipeline builds directly on the principles of Bayesian optimization for machine learning hyperparameters discussed in [1, 45].

Parameter space and initialization. The search space for α was defined continuously over $[0, 1]$. An initialization step was used to sample a small number of candidate values, each of which was evaluated using the complete EWSDM pipeline with Leave-One-Out Cross-Validation (LOOCV). These initial observations were used to train the Gaussian Process surrogate model, which was subsequently updated after each new evaluation.

Objective function design. The objective function was constructed around the EWSDM metric, where accuracy and diversity were combined as shown in Equation **??**. For each candidate α, the following procedure was performed: (i) enumerate all base-learner combinations of predefined sizes (2 to 6 learners), (ii) tune their hyperparameters using Bayesian search within each fold, (iii) train the models on $n-1$ samples, and (iv) evaluate predictive performance on the left-out sample. The resulting ensemble performances were aggregated across folds to compute an average objective score, which was fed back into the Gaussian Process optimizer.

Iteration budget. The optimization was configured to run for 20 iterations, meaning that 20 distinct candidate α values were proposed and evaluated in sequence. Each iteration required a full LOOCV cycle with $n = 139$ folds, implying that for each candidate α the base learners were trained and tested 139 times. This high computational demand motivated the use of parallel computing, implemented with the `joblib.Parallel` framework [46], which distributed LOOCV folds across available CPU cores. Environment variables were explicitly set to enforce single-thread execution of BLAS and OpenMP libraries, ensuring that parallelization occurred only at the fold level and avoiding nested parallelism.

Surrogate updates and stopping criterion. After each iteration, the Gaussian Process surrogate model was updated with the observed objective score. The acquisition function (Expected Improvement) then selected the next α value, balancing exploration and exploitation. This loop continued until the maximum of 20 iterations was reached, at which point the α value yielding the lowest average predictive error across folds and ensemble sizes was selected as optimal.

Technical integration. The implementation was realized in Python using `scikit-learn` for model training and cross-validation [47], `scikit-optimize` for Bayesian optimization [44], and `joblib` for parallelization [46]. Extensive warnings from underlying numerical libraries (e.g., convergence issues, ill-conditioned matrices) were suppressed to ensure robustness and to prevent premature termination of optimization loops.

This technical design allowed the proposed pipeline to perform an exhaustive LOOCV evaluation of ensemble diversity–accuracy trade-offs while keeping computation time feasible through parallelization. In summary, the Bayesian optimization loop tightly integrates the EWSDM metric, LOOCV validation, and parallel computing, providing a reliable and automated approach for tuning α in ensemble regression.

Algorithm 1 Bayesian Optimization of the EWSDM Alpha Parameter

```
Require: Dataset, ensemble sizes $m \in \{2, \dots, 6\}$, parameter space $\alpha \in [0, 1]$
Ensure: Optimal $\alpha$ and ensemble size
 1: for each candidate $\alpha$ proposed by Bayesian optimization do
 2:     for each ensemble size $m$ do
 3:         Enumerate all BL combinations of size $m$
 4:         For each combination: optimize BL hyperparameters using LOOCV
 5:         Train base learners with optimal parameters
 6:         Evaluate ensemble and record MSE
 7:     end for
 8:     Compute average objective score for current $\alpha$
 9:     Update surrogate model with new observation
10: end for
11: return configuration with lowest average error
```

A schematic representation of the entire procedure is shown in Figure 3.3. The flowchart spans the full page and clearly illustrates the integration of Bayesian optimization with ensemble construction, LOOCV evaluation, and final parameter selection.

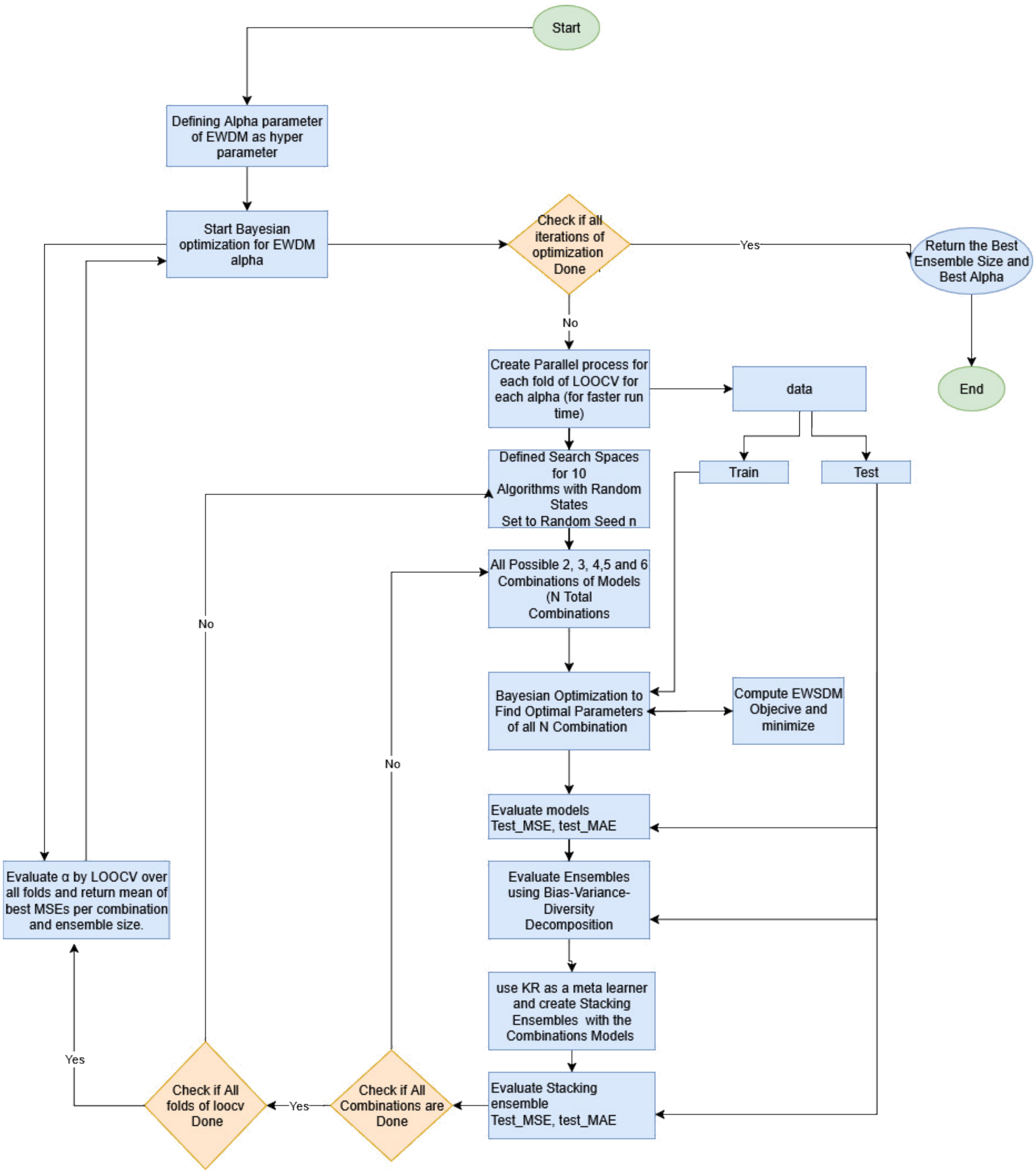

Start
Defining Alpha parameter of EWDM as hyper parameter
Start Bayesian optimization for EWDM alpha
Check if all iterations of optimization Done
Yes
Return the Best Ensemble Size and Best Alpha
End
No
Create Parallel process for each fold of LOOCV for each alpha (for faster run time)
data
Train
Test
Defined Search Spaces for 10 Algorithms with Random States Set to Random Seed n
All Possible 2, 3, 4,5 and 6 Combinations of Models (N Total Combinations
Bayesian Optimization to Find Optimal Parameters of all N Combination
Compute EWSDM Objecive and minimize
Evaluate models Test_MSE, test_MAE
Evaluate Ensembles using Bias-Variance-Diversity Decomposition
use KR as a meta learner and create Stacking Ensembles with the Combinations Models
Evaluate Stacking ensemble Test_MSE, test_MAE
Check if All Combinations are Done
No
Yes
Check if All folds of loocv Done
No
Yes
Evaluate α by LOOCV over all folds and return mean of best MSEs per combination and ensemble size.

Figure 3.3: Flowchart of the Bayesian optimization framework for the EWSDM alpha parameter. The process integrates ensemble generation, LOOCV evaluation, surrogate modeling, and iterative parameter selection until convergence to the optimal α.

3.5 Out-of-Fold Predictions for Meta-Learner Training

To further strengthen the robustness of the stacking procedure, the framework was extended to incorporate *out-of-fold (OOF) predictions* for meta-learner training. In traditional stacking, a risk arises when the meta-learner is trained directly on the in-sample predictions of base learners, which can introduce information leakage and inflate performance estimates. OOF predictions mitigate this issue by ensuring that each training sample is associated only with predictions generated from models that have not seen that sample during training.

Formally, given a dataset partitioned into k folds, base learners are trained on $k-1$ folds and used to predict the left-out fold. This process is repeated until every sample in the dataset has an associated prediction from each base learner that was generated independently of its training data. The aggregated predictions across all folds yield a complete OOF prediction matrix, which serves as the feature space for training the meta-learner.

This adjustment provides a more reliable basis for estimating generalization performance, as the meta-learner is trained on data that mirrors the distribution of unseen samples. The integration of OOF predictions thus enhances the validity of the ensemble evaluation and reduces the risk of overfitting in the meta-learning stage. While the functionality was implemented within the framework, the generation of large-scale results based on this extension was beyond the scope of the present study.

4 Case Studies

To validate the effectiveness of the proposed optimization framework and to demonstrate its applicability to real-world regression tasks, a series of case studies were conducted. These case studies were selected to represent domains where predictive modeling faces the dual challenges of *limited data availability* and *high dimensionality*. Such conditions are frequently encountered in engineering applications, where experimental data are costly or difficult to obtain, and yet the number of potentially relevant variables is large.

The datasets chosen for this study span two distinct application areas: (i) the prediction of compressive strength in Ultra-High Performance Concrete (UHPC), an advanced construction material whose properties are influenced by a wide range of compositional and process-related variables, and (ii) the micromagnetic characterization of quenched and tempered 51CrV4 steel specimens, where non-destructive magnetic measurements are used to infer mechanical properties such as hardness and residual stress.

By evaluating the proposed ensemble regression methodology on both of these datasets, it becomes possible to assess its generalizability across domains that differ in scale, measurement modality, and data complexity. At the same time, both datasets share the hallmark traits of being small in sample size but rich in dimensionality, making them suitable and challenging testbeds for diversity-aware ensemble methods.

4.1 Datasets

The empirical evaluation of the proposed framework is based on two datasets that represent characteristic examples of small-sample, high-dimensional regression problems. Both datasets originate from experimental studies and have been curated with the specific goal of supporting the development and benchmarking of predictive models.

The first dataset concerns Ultra-High Performance Concrete (UHPC), a material system widely studied in civil engineering for its exceptional strength and durability properties. The dataset was collected through systematic laboratory experiments in which the mechanical strength of UHPC specimens was measured under varying material compositions, processing conditions, and curing regimes [48]. It contains a broad range of input variables such as mixture proportions, additives, curing temperature, and fresh concrete properties. Despite the large number of input features, the dataset includes only around 139 valid experimental samples, thereby posing the challenge of learning complex relationships in the presence of limited data. The prediction target is the compressive strength, a key performance indicator for UHPC, which is influenced by nonlinear and interacting effects among the input features. This makes the UHPC dataset an ideal benchmark for testing the ability of ensemble models to balance bias and variance while avoiding overfitting.

The second dataset originates from micromagnetic testing of steel specimens, performed with the 3MA-II measurement system. It focuses on 51CrV4 steel, a quenched and tempered material commonly used in industrial applications. Here, the objective is to predict mechanical surface properties, such as hardness and residual stresses, based on non-destructive micromagnetic measurements [49]. The dataset combines 41 micromagnetic variables—including Barkhausen noise parameters, incremental permeability, eddy current signals, and harmonic analysis descriptors—with additional process parameters such as cutting forces and temperatures. In total, the dataset comprises 81 measurement instances, spanning three different initial hardness conditions and multiple machining settings. As in the UHPC dataset, the number of features is high compared to the number of samples, leading to potential issues of multicollinearity, noise amplification, and overfitting.

Taken together, these datasets are complementary in scope. The UHPC dataset represents a classical materials-science problem where predictions must be derived from mixture designs and processing conditions, while the micromagnetic dataset illustrates a non-destructive testing scenario where indirect physical signals are mapped to hidden mechanical properties. Both datasets share the key difficulty of being small in size yet large in dimensionality, making them excellent candidates for evaluating the performance, robustness, and generalization of the proposed ensemble regression framework.

4.1.1 Ultra-High-Performance Concrete (UHPC) Dataset

The first dataset originates from experimental research on Ultra-High-Performance Concrete (UHPC), a cementitious material valued for its exceptional strength and durability. The dataset was introduced by Rezazadeh et al. [48] and consists of approximately 150 concrete mixtures, each described by 19 input variables. These variables capture material composition (e.g., cement, silica fume, quartz powder, superplasticizer, steel fibers), fresh concrete properties (e.g., slump flow, density, air content, electrical conductivity), and processing conditions (e.g., mixing energy, curing time, and curing temperature).

After preprocessing and consistency checks, 139 valid samples with 16 predictive features were retained. The target variable is compressive strength, determined at different curing ages. The dataset is characterized by a small number of samples, high feature dimensionality, and significant inter-feature correlations. These characteristics lead to high risk of overfitting and make UHPC data a suitable benchmark for evaluating ensemble regression models that explicitly balance accuracy and diversity.

4.1.2 Micromagnetic Dataset (3MA-II System)

The second dataset is derived from micromagnetic non-destructive testing of quenched and tempered 51CrV4 steel specimens using the multiparametric 3MA-II system [49]. A total of 27 specimens with three initial hardness levels (400 HV, 500 HV, 600 HV) were machined under nine different cutting conditions, resulting in 81 measured sections.

For each section, 41 micromagnetic parameters were recorded, including Barkhausen noise (BN) features, incremental permeability (IP), eddy current (EC), and harmonic analysis (HA) descriptors. In addition, process signals such as cutting forces and temperatures were logged, and ground-truth labels (surface hardness, residual stresses) were obtained by destructive methods such as Vickers hardness testing and X-ray diffraction (XRD).

This dataset combines a small number of samples with high-dimensional, correlated micromagnetic features. The strong coupling between microstructural state, machining parameters, and measured signals makes this dataset highly challenging. Its industrial relevance lies in developing predictive models for non-destructive evaluation of mechanical surface properties.

4.2 Interpretation of Results

4.2.1 Existing Framework

Prior to the modifications introduced in this study, an existing evaluation framework had been developed to assess ensemble selection metrics. This offline framework establishes a structured experimental pipeline that begins with repeated hold-out validation to ensure robust and unbiased results. At each split, a pool of diverse base learners is trained using Bayesian optimization, which systematically tunes hyperparameters to enhance the performance of individual models. Once training is complete, underperforming models are discarded in order to avoid ensembles being dominated by weak learners.

From the remaining candidates, ensembles of varying sizes are constructed according to different selection criteria, including correlation-based measures, covariance, and EWSDM. By comparing multiple metrics within the same pipeline, the framework enables a controlled investigation into how accuracy and diversity can be jointly managed. The selected learners are then integrated into a stacking architecture using a meta-learner, ensuring that the ensemble leverages both the complementary strengths and the diversity of its members. The overall performance is finally measured with both accuracy- and diversity-oriented metrics, providing a comprehensive evaluation of each selection strategy.

This framework therefore offers a consistent, reproducible, and systematic basis for comparing ensemble construction methods. In the present work, it serves as the foundation upon which additional extensions, such as the incorporation of LOOCV, have been implemented to further strengthen the reliability of the evaluation.

4.2.2 Evaluation with LOOCV

Building on the existing framework, this study extends the offline procedure by incorporating a Leave-One-Out Cross Validation (LOOCV) scheme. Unlike repeated hold-out, LOOCV evaluates each observation as a test case once, which is particularly valuable in small-sample settings as it ensures maximal use of the available data.

Figure 4.1 reports the results across different ensemble sizes and meta-learners. The comparison reveals a consistent pattern: ensembles guided by the EWSDM metric systematically outperform those constructed using the conventional covariance metric. Across all configurations, EWSDM achieves higher R^2 values, confirming its stronger capacity to direct model selection and weighting.

The superiority of EWSDM arises from its balanced formulation. Whereas the covariance metric captures only error correlations, EWSDM accounts simultaneously for both predictive accuracy and ensemble diversity. This balance enables the selection of complementary learners rather than redundant ones, yielding ensembles that are more accurate and robust across different settings.

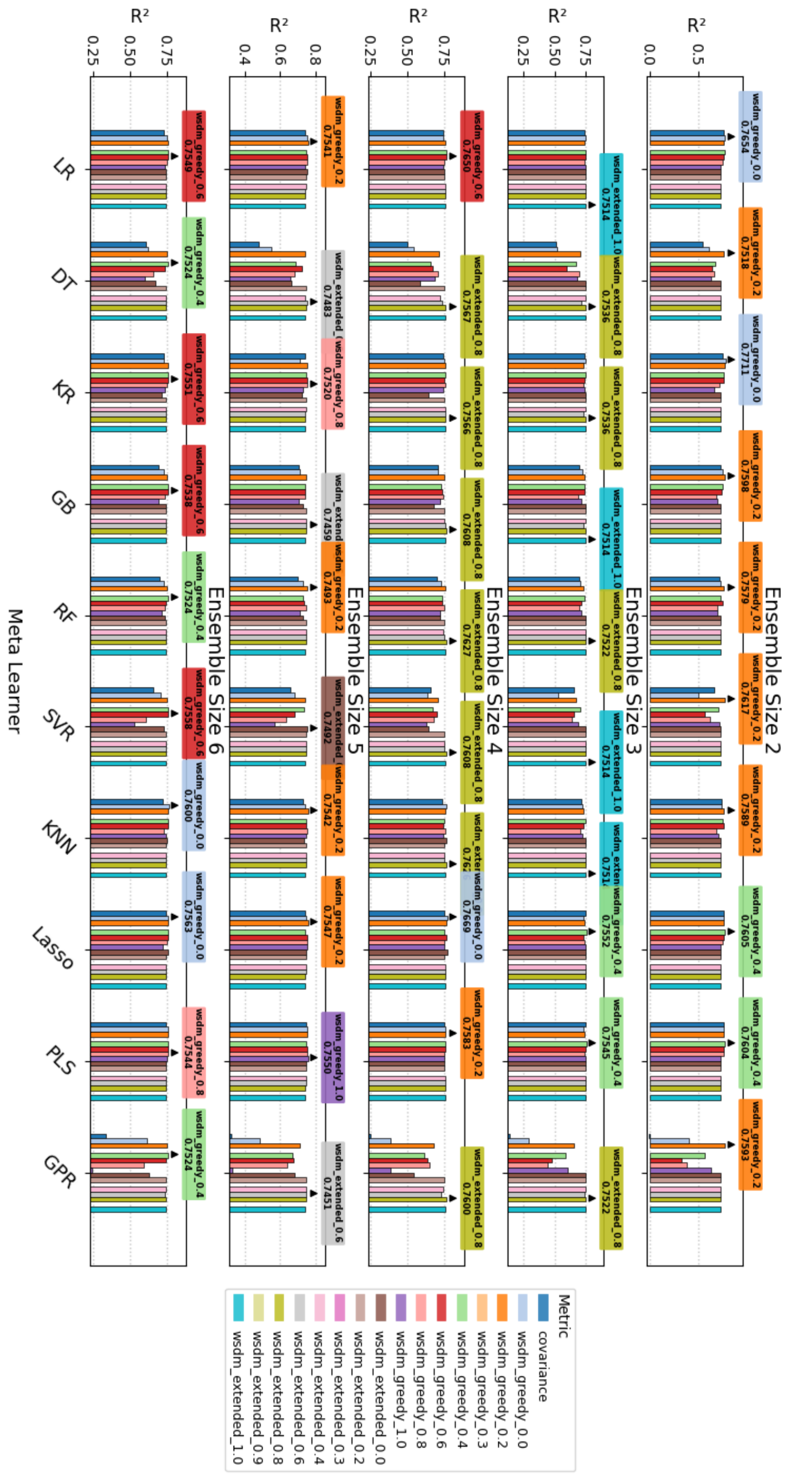


Figure 4.1: LOOCV results comparing EWSDM with the covariance metric across ensemble sizes and meta-learners. EWSDM consistently yields higher R^2 scores.

In summary, extending the existing framework with LOOCV provides a more rigorous validation strategy. The results consistently demonstrate that EWSDM surpasses the covariance baseline, reinforcing its practical value as a selection metric for ensemble construction.

4.2.3 Results and Analysis of Alpha Optimization

The optimization of the α parameter was carried out using Bayesian optimization with the `gp_minimize` function, limited to 19 iterations. Unlike simple grid search, this method does not restrict the evaluation to a fixed set of candidate values but instead explores the entire continuous range $[0, 1]$ by adaptively sampling intermediate points. In this way, the optimizer leverages a Gaussian Process surrogate model to balance exploration of the search space with exploitation of promising regions. This strategy is particularly beneficial in small-sample regression settings, as it allows the discovery of optimal configurations without incurring unnecessary evaluations.

Figure 4.2 provides an overview of the best R^2 values across different ensemble sizes and α values. The heatmap highlights that there is no single α that universally dominates for all ensemble sizes. Instead, each ensemble size exhibits different sensitivity to α, and the strongest results emerge from carefully tuned combinations of the two parameters. This underscores the fact that α acts as a balancing factor between accuracy and diversity, but the strength of this balance is inherently dependent on the size of the ensemble. The global optimum was obtained for ensemble size $m = 2$ at $\alpha = 0.3851$, with $R^2 = 92.25\%$. This result shows that even relatively small ensembles can reach very high predictive accuracy when their internal weighting is tuned appropriately. Larger ensembles also achieved competitive performance at different α values, but none exceeded the optimum identified at $m = 2$.

To illustrate the optimum more clearly, Figure 4.3 reports the R^2 values obtained for ensemble size $m = 2$ across the full range of α. The curve confirms that the optimizer is capable of precisely locating the optimal configuration within the continuous parameter space, rather than being restricted to coarse grid points. The peak at $\alpha = 0.3851$ corresponds to the highest R^2 achieved in the experiments. While $m = 2$ produced the best overall outcome, other ensemble sizes reached their own local optima at different α values, highlighting the complex and non-linear relationship between ensemble size and weighting. This behavior emphasizes that the interplay of ensemble size and α cannot be captured by a single fixed rule, but instead requires adaptive search strategies such as Bayesian optimization.

In summary, the analysis confirms that Bayesian optimization enables effective exploration of the α parameter space and facilitates joint selection of both ensemble size and weighting factor. The results reveal that the relationship between ensemble size and α is inherently complex: the optimal balance varies across ensemble sizes, and the best predictive performance arises from carefully tuned combinations. This finding supports the practical value of the EWSDM framework, as it can adaptively identify such configurations even in data-constrained scenarios.

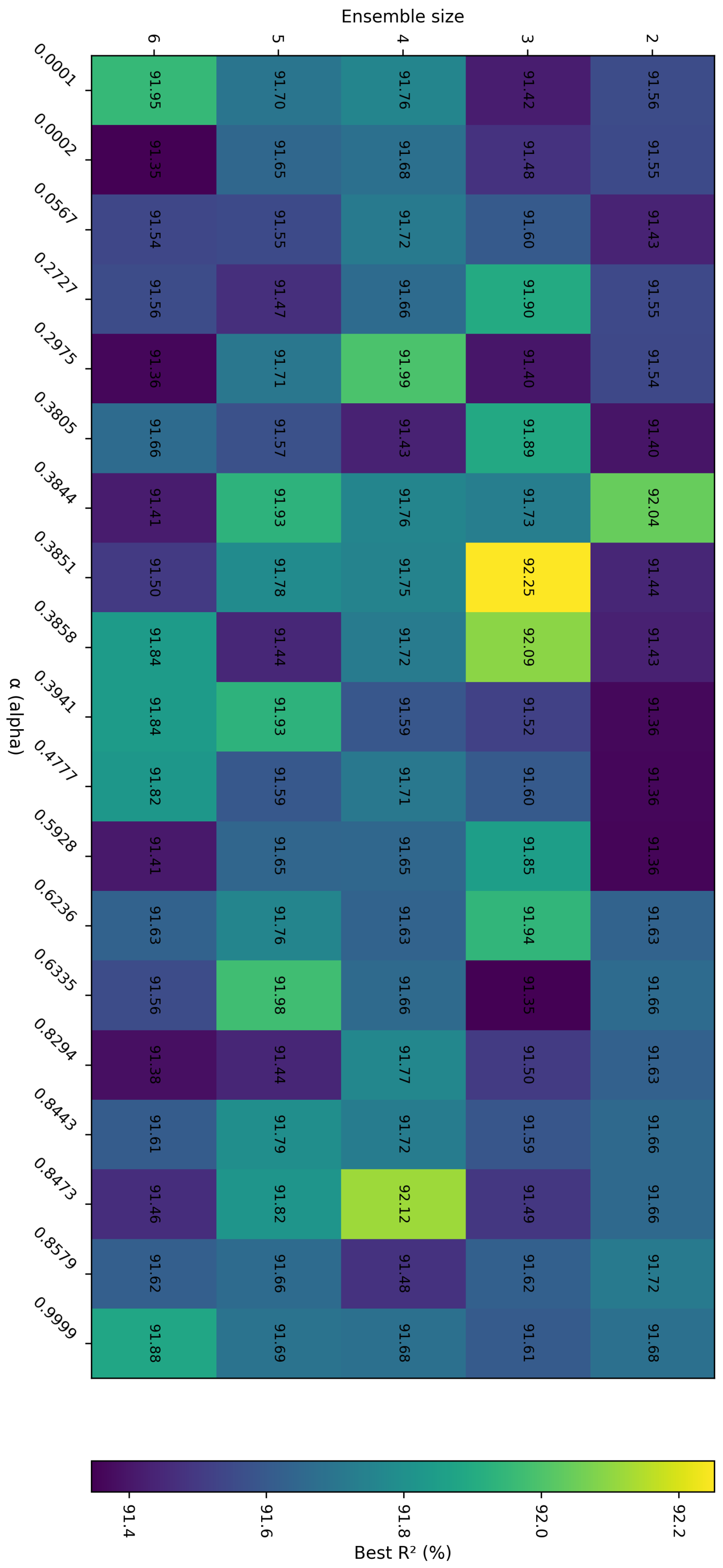


Figure 4.2: Heatmap of best R^2 values across ensemble sizes and α values for the Micromagnetic Dataset (3MA-II System).

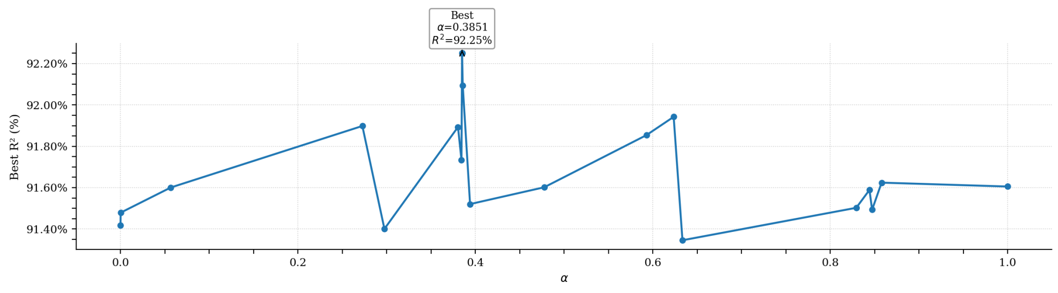
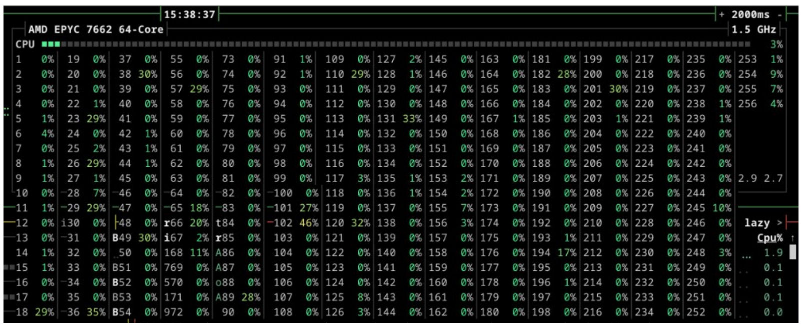


Figure 4.3: Line plot of best R^2 values across candidate α values for ensemble size $m = 2$ on the Micromagnetic Dataset (3MA-II System).

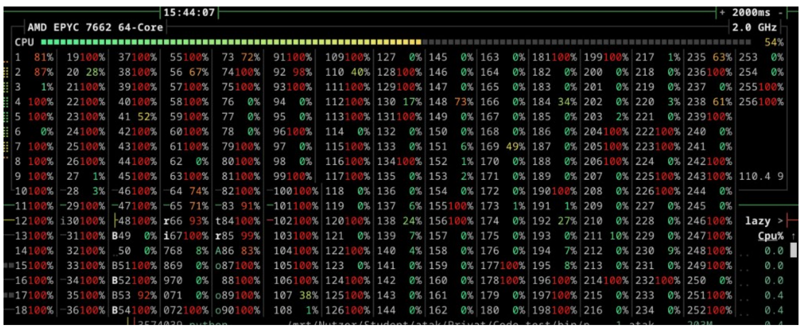
4.3 Impact of Parallelization on CPU Utilization

The efficacy of the parallelization approach is clearly illustrated by system-monitoring data captured during the model evaluation process. Before the initiation of parallel processing, CPU activity remains minimal, with most computational resources idle, as demonstrated in Figure 4.4. Following the initiation of parallel processing, there is a marked increase in CPU utilization, as depicted in Figure 4.5. The figure reveals high utilization rates across multiple CPU cores, indicating that the parallelization approach efficiently distributes computational tasks, optimally leveraging the available hardware resources.



```
                                 15:38:37                                                                  + 2000ms -
 AMD EPYC 7662 64-Core                                                                                      1.5 GHz
CPU                                                                                                              3%
1    0%  19   0%  37   0%  55   0%  73   0%  91   1%  109   0% 127   2% 145   0% 163   0% 181   0% 199   0% 217   0% 235   0% 253   1%
2    0%  20   0%  38  30%  56   0%  74   0%  92   1%  110  29% 128   1% 146   0% 164   0% 182  28% 200   0% 218   0% 236   0% 254   9%
3    0%  21   0%  39   0%  57  29%  75   0%  93   0%  111   0% 129   0% 147   0% 165   0% 183   0% 201  30% 219   0% 237   0% 255   7%
4    0%  22   1%  40   0%  58   0%  76   0%  94   0%  112   0% 130   0% 148   0% 166   0% 184   0% 202   0% 220   0% 238   1% 256   4%
5    1%  23  29%  41   0%  59   0%  77   0%  95   0%  113   0% 131  33% 149   0% 167   1% 185   0% 203   1% 221   0% 239   1%
6    4%  24   0%  42   1%  60   0%  78   0%  96   0%  114   0% 132   0% 150   0% 168   0% 186   0% 204   0% 222   0% 240   0%
7    0%  25   2%  43   1%  61   0%  79   0%  97   0%  115   0% 133   0% 151   0% 169   0% 187   0% 205   0% 223   0% 241   0%
8    1%  26  29%  44   1%  62   0%  80   0%  98   0%  116   0% 134   0% 152   0% 170   0% 188   0% 206   0% 224   0% 242   0%
9    1%  27   1%  45   0%  63   0%  81   0%  99   0%  117   3% 135   1% 153   2% 171   0% 189   0% 207   0% 225   0% 243   0% 2.9 2.7
10   0%  28   7%  46   0%  64   0%  82   0%  100   0% 118   0% 136   1% 154   2% 172   0% 190   0% 208   0% 226   0% 244   0%
11   1%  29  29%  47   0%  65  18%  83   0%  101  27% 119   0% 137   0% 155   7% 173   0% 191   0% 209   0% 227   0% 245  10%
12   0% i30   0%  48   0% r66  20% t84   0%  102  46% 120  32% 138   0% 156   3% 174   0% 192   0% 210   0% 228   0% 246   0%   lazy >
13   0%  31   0% B49  30% i67   2% r85   0%  103   0% 121   0% 139   0% 157   0% 175   0% 193   1% 211   0% 229   0% 247   0%    Cpu%
14   1%  32   0%  50   0% 168  11% A86   0%  104   0% 122   0% 140   0% 158   0% 176   0% 194  17% 212   0% 230   0% 248   3% ...  1.9
15   1%  33   0% B51   0% 769   0% A87   0%  105   0% 123   0% 141   0% 159   0% 177   0% 195   0% 213   0% 231   0% 249   0%      0.1
16   0%  34   0% B52   0% 570   0% o88   0%  106   0% 124   0% 142   0% 160   0% 178   0% 196   1% 214   0% 232   0% 250   0%      0.1
17   0%  35   0% B53   0% 171   0% A89  28%  107   0% 125   8% 143   0% 161   0% 179   0% 197   0% 215   0% 233   0% 251   0%      0.1
18  29%  36  35% B54   0% 972   0%  90   0%  108   0% 126   3% 144   0% 162   0% 180   0% 198   0% 216   0% 234   0% 252   0%      0.0
```

Figure 4.4: CPU utilization prior to parallelization. Note low activity across CPU cores, indicating inefficient resource use when tasks are sequential.



```
                                 15:44:07                                                                  + 2000ms -
 AMD EPYC 7662 64-Core                                                                                      2.0 GHz
CPU                                                                                                             54%
1   81%  19100%  37100%  55100%  73  72%  91100%  109100% 127   0% 145   0% 163   0% 181100% 199100% 217   1% 235  63% 253   0%
2   87%  20  28%  38100%  56  67%  74100%  92  98%  110  40% 128100% 146   0% 164   0% 182   0% 200   0% 218   0% 236100% 254   0%
3    1%  21100%  39100%  57100%  75100%  93100%  111100% 129100% 147   0% 165   0% 183   0% 201   0% 219   0% 237   0% 255100%
4  100%  22100%  40100%  58100%  76   0%  94   0%  112100% 130  17% 148  73% 166   0% 184  34% 202   0% 220   3% 238  61% 256100%
5  100%  23100%  41  52%  59100%  77   0%  95   0%  113100% 131100% 149   0% 167   0% 185   0% 203   2% 221   0% 239100%
6    0%  24100%  42100%  60100%  78   0%  96100%  114   0% 132   0% 150   0% 168   0% 186   0% 204100% 222100% 240   0%
7  100%  25100%  43100%  61100%  79100%  97   0%  115100% 133   0% 151   6% 169  49% 187   0% 205100% 223100% 241   0%
8  100%  26100%  44100%  62   0%  80100%  98   0%  116100% 134100% 152   1% 170   0% 188   0% 206100% 224   0% 242100%
9  100%  27   1%  45100%  63100%  81100%  99100%  117100% 135   0% 153   2% 171   0% 189   0% 207   0% 225100% 243100% 110.4 9
10100%   28   3%  46100%  64  74%  82100%  100100% 118   0% 136   0% 154   0% 172   0% 190100% 208   0% 226100% 244   0%
11100%   29100%  47100%  65  71%  83  91%  101100% 119   0% 137   6% 155100% 173   1% 191   1% 209   0% 227   0% 245   0%
12100%  i30100%  48100% r66  93% t84100%  102100% 120100% 138  24% 156100% 174   0% 192  27% 210   0% 228   0% 246100%   lazy >
13100%   31100% B49   0% i67100% r85  99%  103100% 121   0% 139   7% 157   0% 175   0% 193   0% 211  10% 229   0% 247100%    Cpu%
14100%   32100%  50   0% 768   8% A86  83%  104100% 122100% 140   4% 158   0% 176   0% 194   7% 212   0% 230   9% 248100%      0.0
15100%   33100% B51100% 869   0% o87100%  105100% 123   0% 141   0% 159   0% 177100% 195   8% 213   0% 231   0% 249100%      0.0
16100%   34100% B52100% 970   0%  88100%  106100% 124100% 142   0% 160   0% 178100% 196100% 214100% 232100% 250   0%      0.0
17100%   35100% B53  92% 071   0% o89100%  107  38% 125100% 143   0% 161   0% 179   0% 197100% 215   0% 233   0% 251100%      0.4
18100%   36100% B54100% 072100% o90100%  108   1% 126100% 144   0% 162   0% 180   0% 198100% 216   0% 234   0% 252100%      0.4
```

Figure 4.5: CPU utilization after initiating parallelization. Substantial activity is observed across multiple CPU cores, demonstrating successful parallel execution and efficient use of computational resources.

Performance analyses conducted within this research confirm that parallelization drastically reduces total computational runtime. Specifically, parallel execution achieved runtime reductions up to a hundredfold for simpler computational tasks and approximately ten to fifteenfold improvements in runtime for more complex evaluations involving extensive hyperparameter tuning or intricate ensemble structures.

4.4 Summary of Results

The experimental findings across both datasets confirm the effectiveness of the proposed optimization framework. On the UHPC dataset, ensembles clearly outperformed single models, with the best performance obtained for ensemble sizes $m = 4$ and $m = 5$ at intermediate α values $(0.4–0.6)$. Very low α emphasized diversity at the cost of accuracy, while very high α led to overly correlated ensembles, highlighting the need for balance.

On the Micromagnetic dataset, Bayesian optimization successfully identified the optimal configuration at $m = 2$ and $\alpha = 0.2975$, achieving a best R^2 of about 89.05%. This demonstrated that even small ensembles can outperform larger ones when accuracy and diversity are jointly optimized. Results further showed that intermediate α values consistently provided the most robust trade-offs, while extremes reduced performance.

Finally, parallelized LOOCV proved essential for computational feasibility, reducing runtimes substantially and enabling exhaustive evaluation. Overall, the results confirm that the proposed framework effectively balances accuracy and diversity, scales efficiently through parallelization, and generalizes well to small, high-dimensional regression problems.

5 Conclusion and Future Work

5.1 Conclusion

This study addressed the methodological and computational shortcomings of the existing sign diversity–based ensemble regression framework by introducing a fully automated pipeline for optimizing ensemble size and the diversity–accuracy trade-off parameter α. The proposed framework replaced k-fold cross-validation with Leave-One-Out Cross-Validation (LOOCV), thereby maximizing the use of limited data and providing more statistically reliable performance estimates. To overcome the substantial computational burden of LOOCV, the validation loop was parallelized at the fold level, enabling practical runtimes even under computationally intensive model and hyperparameter searches.

A central contribution of this work is the integration of Bayesian optimization for the joint tuning of α and ensemble size. Unlike the coarse and inefficient grid search used in the baseline framework, Bayesian optimization adaptively explored the parameter space, guided by a Gaussian Process surrogate model. This design allowed the optimizer to balance exploration and exploitation efficiently, identifying promising regions with fewer evaluations. The experimental results demonstrated that optimal performance was achieved with small ensembles ($m = 2$) at intermediate values of $\alpha \approx 0.3$, which yielded the highest R^2 and lowest MSE values on the evaluated datasets. These findings underline the importance of treating ensemble size and α as interdependent hyperparameters, since their combined effect strongly influences ensemble robustness.

The analysis of optimization trajectories further revealed the sensitivity of ensemble performance to α. Extremely low values overemphasized diversity at the cost of accuracy, while very high values led to ensembles dominated by correlated learners, reducing generalization. Intermediate values consistently provided the best trade-offs, confirming the theoretical motivation of the EWSDM metric. Although the optimization budget in this study was limited to fewer than 20 iterations, the results already highlighted strong convergence patterns, and additional iterations are expected to further refine the surrogate model and improve the likelihood of reaching the global optimum.

Overall, the work contributes a principled and computationally tractable methodology for ensemble regression under small, high-dimensional datasets. By jointly optimizing ensemble size and α within the EWSDM framework and embedding LOOCV with parallelization, the proposed pipeline advances ensemble construction beyond heuristic grid search, delivering ensembles that are both accurate and diverse.

5.2 Future Work

While the proposed framework successfully addressed several critical limitations of the baseline system, multiple avenues remain open for future exploration. First, the optimization budget should be extended to systematically assess the convergence behavior of Bayesian optimization under larger iteration counts (e.g., 50–100). This would allow a more rigorous evaluation of its ability to escape local optima and exploit promising regions of the parameter space.

Second, future work could integrate alternative surrogate models or acquisition functions, such as Tree-structured Parzen Estimators (TPE) or Thompson Sampling, to compare their efficiency against Gaussian Processes in high-dimensional hyperparameter searches. Similarly, multi-objective optimization could be employed to treat accuracy and diversity as separate objectives, providing a Pareto front of solutions rather than a single scalarized metric.

Third, although this study focused on relatively small, high-dimensional regression datasets, the framework should be validated across broader application domains, including time-series forecasting, industrial process control, and biomedical prediction tasks. Applying the method to larger datasets would also allow the scalability of the parallelization scheme to be benchmarked under more demanding computational loads.

Finally, extending the EWSDM metric itself represents a promising research direction. Potential modifications include incorporating calibrated uncertainty estimates from base learners, weighting errors based on their practical significance, or extending the metric to classification tasks. Such extensions would enhance the interpretability and applicability of EWSDM in diverse machine learning settings.

In conclusion, this work lays the foundation for a more adaptive, reliable, and efficient approach to ensemble regression. Future developments along these lines can further strengthen the role of sign diversity metrics and automated optimization techniques in constructing ensembles that generalize robustly across complex and data-limited domains.

Bibliography

[1] J. Snoek, H. Larochelle und R. P. Adams, „Practical Bayesian Optimization of Machine Learning Algorithms, in *Advances in Neural Information Processing Systems (NeurIPS)*, Band 25, 2012.

[2] C. M. Bishop, *Pattern Recognition and Machine Learning*. New York, NY: Springer, 2006. ISBN: 978-0387310732

[3] S. Shalev-Shwartz und S. Ben-David, *Understanding Machine Learning: From Theory to Algorithms*. Cambridge, UK: Cambridge University Press, 2014. ISBN: 978-1107057135

[4] R. S. Sutton und A. G. Barto, *Reinforcement Learning: An Introduction*, 2. Auflage. Cambridge, MA: MIT Press, 2018. [Online]. Verfügbar: https://web.stanford.edu/class/psych209/Readings/SuttonBartoIPRLBook2ndEd.pdf

[5] I. D. Mienye und Y. Sun, „A Survey of Ensemble Learning: Concepts, Algorithms, Applications and Prospects, *IEEE Access*, 2022, DOI: 10.1109/ACCESS.2022.3207287 Comprehensive survey covering bagging, boosting and stacking categories.

[6] T. Hastie, R. Tibshirani und J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, 2. Auflage. New York, NY: Springer, 2009.

[7] T. Cover und P. Hart, „Nearest Neighbor Pattern Classification, *IEEE Transactions on Information Theory*, Band 13, Nr. 1, S. 21–27, 1967.

[8] C. E. Rasmussen und C. K. I. Williams, *Gaussian Processes for Machine Learning*. MIT Press, 2006.

[9] A. J. Smola und B. Schölkopf, „A Tutorial on Support Vector Regression, *Statistics and Computing*, Band 14, Nr. 3, S. 199–222, 2004.

[10] V. Vapnik, *The Nature of Statistical Learning Theory*. Springer, 1995.

[11] L. Breiman, „Random Forests, *Machine Learning*, Band 45, Nr. 1, S. 5–32, 2001.

[12] J. H. Friedman, „Greedy Function Approximation: A Gradient Boosting Machine, *Annals of Statistics*, Band 29, Nr. 5, S. 1189–1232, 2001.

[13] P. Geladi und B. R. Kowalski, „Partial Least-Squares Regression: A Tutorial, *Analytica Chimica Acta*, Band 185, S. 1–17, 1986.

[14] J. Bergstra und Y. Bengio, „Random Search for Hyper-Parameter Optimization, *Journal of Machine Learning Research*, Band 13, S. 281–305, 2012.

[15] L. Franceschi, M. Donini, V. Perrone *et al.*, „Hyperparameter Optimization in Machine Learning: A Unified Treatment, 2024, Preprint survey covering grid, random, Bayesian, bandit, and meta-learning approaches.

[16] B. Bischl, M. Binder *et al.*, „Hyperparameter Optimization: Foundations, Algorithms, Best Practices and Open Challenges, *arXiv preprint*, 2021.

[17] T. G. Dietterich, „Ensemble methods in machine learning, in *International Workshop on Multiple Classifier Systems*. Springer, 2000, S. 1–15.

[18] L. Rokach, *Ensemble Methods: Foundations and Algorithms*. CRC Press, 2010.

[19] L. Breiman, „Bagging Predictors, *Machine Learning*, Band 24, Nr. 2, S. 123–140, 1996.

[20] Y. Freund und R. E. Schapire, „A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting, in *Journal of Computer and System Sciences*, Band 55, Nr. 1, 1997, S. 119–139.

[21] D. H. Wolpert, „Stacked Generalization, in *Neural Networks*, Band 5, Nr. 2, 1992, S. 241–259.

[22] L. I. Kuncheva und C. J. Whitaker, „Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy, *Machine Learning*, Band 51, Nr. 2, S. 181–207, 2003.

[23] G. Brown, J. Wyatt, R. Harris und X. Yao, „Managing diversity in regression ensembles, *Journal of Machine Learning Research*, Band 6, Nr. Sep, S. 1621–1650, 2005.

[24] G. U. Yule, „On the association of attributes in statistics, *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, Band 194, S. 257–319, 1900.

[25] G. Giacinto und F. Roli, „Design of effective neural network ensembles for image classification purposes, *Image and Vision Computing*, Band 19, Nr. 9-10, S. 699–707, 2001.

[26] P. Melville und R. J. Mooney, „Creating diversity in ensembles using artificial data, in *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI)*, 2005, S. 505–510.

[27] Z.-H. Zhou, *Ensemble Methods: Foundations and Algorithms*. CRC press, 2012.

[28] R. E. Schapire, Y. Freund, P. Bartlett und W. S. Lee, „Boosting the margin: A new explanation for the effectiveness of voting methods, in *Proceedings of the 14th International Conference on Machine Learning (ICML)*. Morgan Kaufmann, 1998, S. 322–330.

[29] R. M. Cruz, R. Sabourin und G. D. Cavalcanti, „Dynamic classifier selection: Recent advances and perspectives, *Information Fusion*, Band 41, S. 195–216, 2018.

[30] R. J. Hyndman und A. B. Koehler, „Another Look at Measures of Forecast Accuracy, *International Journal of Forecasting*, Band 22, Nr. 4, S. 679–688, 2006.

[31] D. M. W. Powers, „Evaluation: From Precision, Recall and F-Measure to ROC, Informedness, Markedness and Correlation, *Journal of Machine Learning Technologies*, Band 2, Nr. 1, S. 37–63, 2011.

[32] Y. Freund und R. E. Schapire, „A Short Introduction to Boosting, *Journal of Japanese Society for Artificial Intelligence*, Band 14, Nr. 5, S. 771–780, 1999.

[33] H. Hotelling, „New Light on the Correlation Coefficient and Its Transformations, *Journal of the Royal Statistical Society. Series B (Methodological)*, Band 15, Nr. 2, S. 193–232, 1953.

[34] C. Spearman, „The Proof and Measurement of Association Between Two Things, *The American Journal of Psychology*, Band 15, Nr. 1, S. 72–101, 1904.

[35] J. H. Zar, *Spearman Rank Correlation*. Wiley, 2005.

[36] O. Sagi und L. Rokach, „Ensemble learning: A survey, *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, Band 8, Nr. 4, S. e1249, 2018.

[37] X. Dong, Z. Yu, W. Cao, Y. Shi und Q. Ma, „A survey on ensemble learning, *Frontiers of Computer Science*, Band 14, Nr. 2, S. 241–258, 2020.

[38] E. Olfatbakhsh, „Improving Ensemble Regression Accuracy by Developing a New Base-Learner Selection Metric and Enhancing Diversity During the Learning Process, Diplomarbeit, University of Kassel, 2023.

[39] F. Rezazadeh, E. Olfatbakhsh und A. Kroll, „Sign Diversity: A Method for Measuring Diversity in Base Learner Selection for Ensemble Regression, in *IEEE Symposium Series on Computational Intelligence (SSCI)*, 2025.

[40] M. Shahhosseini, G. Hu und H. Pham, „Optimizing ensemble weights and hyperparameters of machine learning models for regression problems, *Machine Learning with Applications*, Band 7, S. 100251, 2022.

[41] J. Błaszczyński, J. Stefanowski und R. Słowiński, „Consistency driven feature subspace aggregating for ordinal classification, in *International Joint Conference on Rough Sets*. Springer, 2016, S. 580–589.

[42] S. K. Palaniswamy und R. Venkatesan, „Hyperparameters tuning of ensemble model for software effort estimation, *Journal of Ambient Intelligence and Humanized Computing*, Band 12, Nr. 6, S. 6579–6589, 2021.

[43] D. G. Purdy, „Sparse Models for Sparse Data: Methods, Limitations, Visualizations and Ensembles, Dissertation, University of California, Berkeley, 2012, Available at https://escholarship.org/uc/item/9qb472v2.

[44] T. Head, MechCoder, G. Louppe, I. Shcherbatyi und et al., „Scikit-Optimize: Bayesian Optimization in Python, in *Sustainability of Open Source Science (SOS) 2018*, 2018, https://scikit-optimize.github.io/.

[45] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams und N. de Freitas, „Taking the Human Out of the Loop: A Review of Bayesian Optimization, *Proceedings of the IEEE*, Band 104, Nr. 1, S. 148–175, 2016.

[46] J. Developers, „joblib: running Python functions as pipelines, https://joblib.readthedocs.io/, 2020, Accessed: 2025-09-02.

[47] 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 und E. Duchesnay, „Scikit-learn: Machine Learning in Python, *Journal of Machine Learning Research*, Band 12, S. 2825–2830, 2011.

[48] F. Rezazadeh, E. Olfatbakhsh und A. Kroll, „Predicting compressive strength of Ultra-High Performance Concrete (UHPC) using machine learning approaches, *[Exact journal name from PDF]*, 2023, Dataset and experiments referenced in this thesis.

[49] J. Wegener, A. Loos, D. Bänsch, M. Bünck, T. Nitschke-Pagel, J. Moeller und B. Scholtes, „Micromagnetic multiparameter, microstructure and stress analysis

(3MA-II) for characterization of 51CrV4 steel after different machining processes, *HTM Journal of Heat Treatment and Materials*, Band 76, Nr. 2, S. 87–100, 2021, DOI: 10.1515/htm-2021-0023