Østfold University College

Flexible Ensemble Structures for Gradient Boosting

Master Thesis in Applied Computer Science

Ole-Edvard Ørebæk

School of Computer Sciences
Østfold University College
Halden
August 15, 2021

Abstract

Hyperparameters are essential to the predictive performance of machine learning models and require unique configurations to best fit a given task. The notion of optimizing hyperparameters for prediction performance is referred to as hyperparameter optimization. Hyperparameters are typically handled as one set of values used universally in the training procedure of machine learning models. However, certain complex model types require multiple sets of hyperparameters based on contained components. For instance, this is relevant for artificial neural networks, which have both global hyperparameters that affect the entire network and per-layer hyperparameter that define aspects relevant to each layer. Both the global and per-layer hyperparameters in this context are essential to neural networks' predictive performance. Gradient boosting algorithms are another type of machine learning method that produces complex models in the form of decision tree ensembles. In such ensembles the contained decision trees work together to make predictions by directly compensating for each other inaccuracies. The hyperparameters of gradient boosting algorithms are handled in the typical way where all trees are defined by the same set of hyperparameters. However, it seems reasonable to theorise that such ensembles could benefit from per-tree hyperparameters, considering that each tree can be conceptualized to fit individual tasks.

In this thesis we define per-tree hyperparmeters for gradient boosting ensembles as the term "flexible ensemble structures", and propose two approaches to their optimization. These are respectively named Holistic and Incremental flexible ensemble structure optimization. We investigate the application of flexible ensemble structures through 5 experiments based on XGBoost ensembles of 5 trees. Specifically, we focus on their benefit to prediction performance, determine how applicable they are, based on their optimization difficulty, and investigate how they can be effectively optimized. From the results of the experiments, we find that flexible ensemble structures seem significantly beneficial, based on the fact that they considerably outperformed traditional structures in terms of prediction performance while remaining manageable in optimization difficulty. In fact, we find indications that in certain scenarios it is practically easier to obtain good prediction performance with flexible ensemble structures than with traditional ones. Comparing the proposed optimization approaches, we find that the Holistic approach was clearly more effective, and we suggest this as the standard for flexible ensemble structure optimization. Beyond this, we find several aspects with the potential to be exploited for increased optimization effectiveness.

Acknowledgments

First and foremost, I would like to thank my supervisor, Marius Geitle, for his valuable guidance, insights and support throughout my work on the thesis. I would also like to thank my mother and father, who have been patient and supportive regarding the time required to finish the assignment.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In machine learning algorithms, *hyperparameters* is the term for parameters who's values, among other things, define the architecture of the model(s) produced by the algorithm. Due to their nature, tuning algorithms' hyperparameters can significantly impact produced models' prediction performance for better or worse. Additionally, the hyperparameters of a given machine learning method affects prediction performance differently for each dataset, and therefore need to be configured uniquely for a given task to reach optimal prediction performance. This makes tuning hyperparameters to optimize prediction performance, referred to as hyperparameter optimization, an absolutely essential part of most machine learning tasks [64, 3].

Hyperparameters are typically input-parameters to the algorithm itself, and are usually defined as one static set of values used universally in all aspects of the training procedure [64]. This is inherently fine for algorithms that produce simple models, like singular decision trees, but can be sub-optimal for ones that produce more "complex" models. This can be demonstrated with artificial neural networks, which can have both "global hyperparameters" that affect the entire network, and "per-layer hyperparameters," which define aspects of individual layers. The number of hyperparameters for neural networks therefore depends on the number of hidden layers. More importantly; The prediction performance of neural networks is very sensitive to the values of their per-layer hyperparameters [16, 2]. Per-layer hyperparameters are thus absolutely essential to the prediction performance of artificial neural networks, and should not be excluded from optimization processes.

Gradient boosting decision tree algorithms are types of algorithms that also produces complex models and are widely used due to their effectiveness in both classification and regression tasks, general versatility, and potential to be paralellized [12]. Gradient boosting algorithms are based on boosting, which is the notion of combining many iteratively created trees, with weak individual performance, into an ensemble of trees with strong performance [19]. Gradient boosting takes this principle a step further by treating the task of iteratively adding trees like a gradient descent problem in functional space [12]. While this is already very effective as it allows the algorithm to make the iteratively created trees compensate for the inaccuracies of the previously added components, there is still an aspect which potentially could be improved; All ensemble trees are defined by the same hyperparameter configuration. Because each ensemble tree is generated to compensate for the inaccuracies of the earlier trees [24], they can be conceptually thought of as fitting to individually unique tasks. And as we already know that hyperparameters achieve best prediction performance when they are fine tuned to a given task [3], it seems reasonable to theorize

that the prediction performance of the finalized ensemble could benefit from each of its contained trees having unique hyperparameter configurations. However, because much of the model architecture relevant to each component is defined universally by the same set of hyperparameters, this is currently not the case. Thus, investigating the effects of per-tree hyperparameter configurations for gradient boosting ensemble trees could be worthwhile.

1.1 Research Questions and Method

The potential benefits of per-tree hyperparameters are largely determined by how they affect prediction performance, as this is the primary benefit of hyperparameter optimization in general [64]. However, regardless of how beneficial they are to prediction performance, the application of per-tree hyperparameter configurations can depend on other factors. One such factor is the difficulty of optimization. Each hyperparameter added to an optimization process exponentially increases the number of unique configurations. This phenomena is often referred to as the curse of dimentionality [64]. Due to the curse of dimentionality, it is possible that the difficulty of optimizing per-tree hyperparameters with gradient boosting scales similarly. If this is the case, the application of per-tree hyperparameters could quickly become unreasonable, due to extreme requirements for computational resources. Beyond these aspects, we do not know how per-tree hyperparameters for gradient boosting ensembles are best optimized, of which insight is necessary to standardize and effectivize their use. Thereby, we define the following research questions:

RQ1: *To what extent are per-tree hyperparameters beneficial for the prediction performance of gradient boosting ensembles?*

RQ2: *To what extent are per-tree hyperparameters detrimental to the optimization difficulty of gradient boosting ensembles?*

RQ3: *How can gradient boosting ensembles with per-tree hyperparameters be effectively optimized?*

1.1.1 Method

To answer the research questions, we first need a solid understanding of hyperparameter optimization and gradient boosting algorithms. We also need insights into concepts in other contexts of machine learning similar to per-tree hyperparameters with gradient boosting ensembles. The obtained knowledge will provide a foundation for how we explore and evaluate the effects of per-tree hyperparameter for gradient boosting ensembles.

Because per-tree hyperparmaeters for gradient boosting ensembles is a novel concept, we need to clearly define it in relation to how the hyperparameters of gradient boosting ensembles are traditionally handled. We also need to establish a few approaches for how they can be optimized. Thereafter, we will need to obtain relevant insights surrounding the research questions by conducting a set of experiments based on gradient boosting ensembles with per-tree hyperparameters. Because we do not know how the inclusion of per-tree hyperparameters affects prediction performance, we will investigate relatively small ensembles of 5 trees. This way we can (hopefully) keep the search difficulty manageable, while still optimizing enough ensemble trees to investigate the effects of the per-tree hyperparmeters.

Based on the results of the experiments, we will provide an answer to each of the presented research questions and outline their relevancy in the context of machine learning research.

1.2 Outline

This section provides an overview of the remaining chapters of the thesis.

Chapter 2 provides background information relevant to the thesis. The chapter begins by providing an overview of the hyperparameter optimization problem and outline some common methods for this task. Thereafter, the concepts of boosting and gradient boosting are explained, and some well known and used machine learning algorithms based on these concepts are abstractly described. Lastly, a collection of earlier research relevant to the thesis topic is discussed.

Chapter 3 defines per-tree hyperparameters for gradient boosting ensembles as the term "flexible ensemble structures", and explains it in relation to how gradient boosting ensembles' hyperparameters are typically handled. Furthermore, two approaches to how flexible ensemble structures can be optimized are proposed.

Chapter 4 outlines the groundwork for the experiments of the thesis. This includes the selection of datasets, gradient boosting algorithm, hyperparameters to optimize, hyperparameter optimization methods, and hardware used for computations.

Chapter 5 defines the thesis experiments and describes their individual methodologies.

Chapter 6 presents the results of each experiment.

Chapter 7 provides a discussion around the thesis. The research questions are here restated and answered based on the experiment results, the relevancy of the results are discussed in relation to earlier research, and the limitations of the thesis are acknowledged.

Chapter 8 outlines some implications for future work based on the limitations of the thesis. Some additional ideas for how flexible ensemble structures could be handled and modified are also discussed.

Chapter 9 wraps up the thesis by summarizing its contents before providing a clear conclusion.

Chapter 2

Background

In this chapter, we outline the background and related work of thesis. The chapter is divided into three sections. In Section 2.1, we explain the concept of hyperparameter optimization, and outline some of the most common techniques for this task. Specifically; Grid Search, Random Search, Evolutionary Algorithms, and Bayesian Optimization. In Section 2.2, we define the concepts of boosting and gradient boosting, and abstractly describe some well known and used machine learning algorithms based on these concepts: AdaBoost, XGBoost, LightGBM, and CatBoost. Finally, in Section 2.3, we outline and discuss earlier research relevant to the topic of the thesis.

2.1 Hyperparameter Optimization

Machine learning algorithms typically have two types of parameters. The first type is referred to as *model parameters*, being internal parameters native to produced models that are automatically tuned by the algorithm during the training procedure. The other type of parameters, however, are not automatically tuned during training, but on the contrary, help define how the training procedure is to be executed and/or define the architecture of the models to be produced. These types of parameters are named *hyperparameters* and are usually set as input parameters of the algorithm itself [64]. Given their nature, the values of the algorithm's hyperparameters can greatly impact the performance of produced models. Thus, tuning algorithms' hyperparameters to optimize model performance is an essential part of machine learning, and is usually referred to as *hyperparameter optimization* [3].

However, hyperparameter optimization is not trivial. Optimal hyperparameter configurations for a given machine learning algorithm varies greatly based on the dataset used, and different machine learning algorithms typically have different hyperparameters altogether. These factors, coupled with the fact that there is no consistent way to predict good hyperparameter configurations, encourages the use of *black box* methods [2, 64, 25]. These types of methods are typically based on approaches similar to "trial and error, though they can be much more sophisticated than this description would imply. Some of the most common methods to hyperparameter optimization are outlined in the sections below. Namely; Grid Search, Random Search, Evoluationary Algorithms and Bayesian Optimization.

2.1.1 Grid Search

Grid Search is one of the most simple and commonly used methods to hyperparameter optimization. It finds hyperparameter configurations by brute-forcing all possible combinations of a defined grid of values [64]. However, Grid Search is generally considered a very inefficient method due to its exhaustive search approach, especially when optimizing many hyperparameters. This is because the amount of unique configurations increases exponentially with each hyperparameter to optimize. Exploring the entire search space with Grid Search thus quickly becomes extremely computationally expensive [64]. Grid Search also suffers from wasting resources on exploring uneventful areas of the search space [2].

2.1.2 Random Search

Random Search is another simple and widely used approach to hyperparameter optimization. As implied by its name, Random Search optimizes hyperparameters by evaluating random configurations based on lower and upper value bounds for each hyperparameter. Despite being just as simple in implementation as Grid Search, it is significantly more efficient. This is because different hyperparameters have different levels of impact on prediction performance. Compared to Grid Search, Random Search therefore spends less computational resources on optimizing non-effective hyperparameters [2, 64]. Random Search was additionally made more efficient by Geitle and Olsson [25], who added functionality for adaptively narrowing the search space as the number of iterations increases, while keeping the method simple in implementation.

2.1.3 Evolutionary Algorithms

Evolutionary Algorithms is an umbrella-term for search algorithms inspired by principles found in natural evolution [46, 35]. According to Charles Darwin [14], natural evolution in the animal kingdom is determined by the following aspects: The variation of characteristics between parents and child, the heritability of characteristics, and a competitive environment that ensures that only the fittest individuals can survive and reproduce. These are in turn what forms the basis for the functionality of Evolutionary Algorithms.

Evolutionary Algorithms optimize a given objective by producing sets of solutions, commonly referred to as populations of individuals, where each iteration is called a generation. For each generation, a set of new individuals are created by in some way reproducing the individuals of the previous generation. Some of the most common methods of reproducing individuals, are crossover and mutation. Crossover is typically the primary method of reproduction, and works by producing a new individual by combining the characteristics of multiple individuals from the previous generation. Mutation is generally used as a secondary method to reproduction, usually in combination with crossover, and works by in some what altering certain aspects of singular individuals. Both of these methods are typically stochastic to ensure that produced individuals are different from their parents. To determine the "fitness" of the individuals, each one is evaluated on a fitness function, which in the context of optimization tasks, is equivalent to the objective function. Reproduction is often prioritized for individuals of higher fitness, to explore the well performing areas of the search space [46].

There are many different Evolutionary Algorithms, each with their own strategies and optimal application areas. To name a few: Genetic Algorithms [28], Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [26], Differential Evolution [58], Success-History based Adaptive Differential Evolution (SHADE) [59], and Particle Swarm Optimization [34] are well known, and widely used examples.

2.1.4 Bayesian Optimization

Abstractly speaking, Bayesian Optimization algorithms are procedures that iteratively optimizes a given objective through points of evaluation, each based on the results of the previous evaluations. This is achieved through a sophisticated method utilizing two components: a surrogate model and an acquisition function. These components have a cyclic relation ship. The surrogate model attempts to predict results on the objective, based on the observed results of earlier evaluations, and the acquisition function uses the surrogate model to determine the next point of evaluation. New points of evaluation are determined based on balancing two factors. One factor is to exploit the surrogate model's predictions for promising areas of the search landscape, the other is to explore areas currently not covered. By balancing these two factors, the acquisition function attempts to find increasingly better results, while ensuring that the algorithm does not converge to a local optima, by routinely exploring new areas of the search landscape [64].

The general procedure of Bayesian Optimization algorithms on hyperparmeter optimization problems can thus be outlined as follows [64]:

1. Create an initial surrogate model.
2. Use the acquisition function to determine a hyperparameter configuration to evaluate based on the predictions of the surrogate model, while balancing the factors of exploitation and exploration.
3. Evaluate the selected hyperparamter configuration on the objective function.
4. Update the surrogate model with the results of the evaluated hyperparameter configuration.
5. Repeat steps 2 to 4 for the number of iterations specified by the user.

Generally speaking, there are three types of Bayesian Optimization algorithms, based on the type of surrogate model used [64]: Gaussian Process [55], Random Forest [30], and Tree Parzen Estimator [2]. For each of these types, there are frameworks frameworks that contain Bayesian Optimization implementations. For instance, for the Gaussian Process surrogate model type, there are Spearmint [57], BayesOpt [40], Scikit-optimize [39], GpFlowOpt [36], and Sherpa [27]. For the Tree Parzen Estimator type, there are Hyperopt [4], BOHB [17], Optunity [13], and Osprey [41]. Random Forest seems to be the least implemented surrogate model type, but SMAC [38] is a good example.

2.1.5 Method Comparisons

Generally, Random Search is considerably more efficient than Grid Search for optimizing hyperparameters. However, as these methods typically do not have abilities of adapting the search space, with some exceptions like Adaptive Random Search [25], they are both

considered inefficient compared to Evolutionary Algorithms and Bayesian Optimization. Each of these two methods have their benefits and drawbacks. Evolutionary Algorithms are often relatively simple in search approach and easy to parallelize, but can require a large amount of iterations to find optimal configurations. Comparably, Bayesian Optimization methods typically find good hyperparameter configurations in less iterations, but are based on complex search approaches that are computationally demanding and hard to parallelize [64].

2.2 Boosting

Boosting refers to a general machine learning principle based on producing accurate predictive models by creating an ensemble of comparatively inaccurate models [20]. The basis for boosting originates from the "PAC" learning model [33], a theoretical framework for studying machine learning. With this learning model, Kearns and Valiant [32] introduced the term *weak learner*, being defined as a model that have predictive capabilities only slightly better than random guessing. Kearns and Valiant also raised the question of whether weak learners could be "boosted" into a strong learner with an arbitrarily high accuracy to the training data. This in turn inspired the first proposed boosting algorithms [54, 19] which were based on this very theory.

A common factor of how the ensemble of weak learners is created in boosting algorithms is that each iterativly added weak learner is made to compensate for the inaccuracy of the previously added weak learners. How the weak learners are made to compensate, however, varies with different boosting methods and algorithms. Different methods and a few selected boosting algorithms; AdaBoost, XGBoost, LightGBM and CatBoost, are outlined below.

2.2.1 AdaBoost

AdaBoost (short for Adaptive Boosting), proposed by Freund and Schapire [21], is generally considered to be the first practical boosting algorithm [20, 42]. The goal of AdaBoost is to reach a final model which has low error, relative to the training data, after a certain amount of iterations. To accomplish this, the algorithm generates for each iteration a weak learner that is added to an ensemble. The final model then becomes this ensemble of a number of weak learners, equal to the number of iterations in the algorithm, where final predictions are calculated based on a weighted majority vote [21].

As mentioned in Section 2.2, in boosting algorithms, the iteratively added weak learners are made to compensate for the inaccuracies of the previously added weak learners. AdaBoost handles this through the use of *weights*. Weights are numbers attached to each instance in the training data, which represent how much individual training instances should be taken into account when generating weak learners. Higher weights corresponds to higher influence how the weak learner is generated. The weights are initially set to be equal for all data instances, and are for each iteration in the algorithm updated. Updates to the weights are based on the performance of the previous iteration's generated weak learner, and are tweaked so that inaccurately predicted instances of the previous iteration are weighted higher in the current iteration. In this way the weak learners are generated adaptively based on the results of previous iterations. This is in turn where the name, AdaBoost, is derived from [21].

2.2.2 Gradient Boosting Decision Tree Algorithms

Friedman [23] introduced in 2001 the principle of treating additive expansion of trees as steepest descent minimization. In this principle decision trees that best minimize error for a given iteration in the boosting procedure are regarded as steps towards the steepest descent. In other words; Tree boosting is practically treated as a gradient descent problem in functional space. Friedman also incorporated the notion of *shrinkage* into the gradient boosting principle, being a regularization method. Regularization methods are designed to prevent overfitting by constraining the fitting procedure [23]. Shrinkage is, in this context, defined as scaling each update of the boosting model by the value of a learning rate parameter. Gradient boosting algorithms thereby have a minimum of two regularization parameters, being the *learning rate* and the *number of components* (iterations), which a natural parameter given that gradient boosting is a form of additive expansion.

Both the learning rate and the number of components can individually affect the optimial value of the other. Interestingly, Friedman found, in relation to this, that lower learning rate values result in better performance. However, this comes at the cost of requiring a higher number of components, thus resulting in longer training times and larger, more complex models [23]. This trade-off can often be a great inconvenience when working with machine learning problems, especially when training on very large datasets, or when using gradient boosting algorithms in real-time systems.

Stochastic gradient boosting can be abstractly explained as elements of randomness added into the gradient boosting procedure [24]. Specifically, it is the gradient boosting algorithm modified to a minor degree by, in each iteration, randomly drawing a subsample (without replacement) from all training instances and using this subsample, as opposed to the full training set, to fit the weak learner [24]. Fitting models to drawn subsamples of data instances is referred to as "bagging" and was first introduced by Breiman [6] in 1996. Breiman further extended on this idea in 1999 by proposing "adaptive bagging" [7], being a combination of boosting and bagging, where the regular weak learners of boosting algorithms where replaced with bagged weak learners. This was in turn the primary inspiration behind Stochastic Gradient Boosting.

2.2.3 XGBoost

XGBoost, proposed by Chen & Guestrin [12], is an open source gradient boosting decision tree algorithm that is widely considered state-of-the-art, as it has frequently been featured in winning models of machine learning competitions, hosted on websites such as Kaggle [1]. One of the main attractions of XGBoost is that it is designed to be scalable to as many scenarios as possible. This is achieved as a result of several implemented innovations compared to earlier boosting algorithms. The primary innovations are; the ability to handle sparse data; a proposed theoretically justified weighted quantile sketch for efficient proposal calculation; the algorithm is made to be parallelizable through a sparsity-aware algorithm; and the algorithm exploits out-of-core computation, which makes it very efficient. The algorithm also makes minor improvements to the regularized learning objective, compared to Friedman's [22] earlier proposal. Specifically, XGBoost uses shrinkage of weights added per iteration to regulate the influence of individual trees, as well as random subsampling of columns (features) and rows to avoid overfitting, as seen in, e.g., RandomForest [8]. XGBoost also uses an "approximate greedy algorithm" for finding the best feature split,

[1] `https://www.kaggle.com/`

which is designed to be less computationally demanding than an "exact greedy algorithm" [12]. However, it could still be argued that efficency and scalablility is unsatisfactory in the split-finding aspects when working with large datasets due to having to scan every instance in order to calculate the split information gains [31].

2.2.4 LightGBM

With the proposal of LightGBM [31], the authors set out to improve on the implementations of earlier gradient boosting decision tree algorithms, XGBoost included. Specifically, LightGBM attempts to improve on earlier implementations' approach to split-finding. The changes proposed with LightGBM are based on the idea of reducing the number of instances and features of a given dataset, and can be summarized into two techniques; *Gradient-based One-Side Sampling* (GOSS) and *Exclusive Feature Bundling* (EFB) [31].

GOSS utilizes the notion that data instances with different gradients affect the computation of information gain differently. Specifically that instances with larger gradients contribute more to the information gain [31]. Thus when the data is down-sampled, instances with larger gradients should be kept to retain accuracy of information gain estimation, and instances with smaller gradients should be randomly dropped. The authors proved that this can give better accuracy of information gain estimation compared to uniform random sampling [31].

EFB, on the other hand, exploits the observation that many datasets have a large number of features which are often times quite sparse. This means that the number of effective features can potentially be reduced nearly losslessly. An example of this is datasets where many of the features are independent of each other (exclusive), similar to resulting datasets of executing one-hot encoding. Feature Bundling of the independent features can therefore be assimilated to a graph coloring problem. The graph coloring problem is in this context based on feature defined vertices with edges between every non-mutually exclusive feature, which is solved with a greedy algorithm with constant approximation ratio [31].

Compared to other gradient boosting methods like XGBoost, the innovations proposed with LightGBM led to it having significantly faster training times, while achieving competitive prediction performance [31].

2.2.5 CatBoost

CatBoost (short for Categorical Boosting) [48] is another open-source gradient boosting decision tree library that technically outperforms both XGBoost and LightGBM. The authors discovered a problem present in all previously implemented gradient boosting algorithms. Namely, what they call a *prediction shift*, which is caused by a certain type of target leakage. This prediction shift comes as a result of a prediction model, obtained after a number of boosting iterations, being reliant on the training instances' targets, which further leads to a shift in the distribution of training instances from the distribution of the test instances. Prediction shifts are bad because they negatively affects the model's generalization ability, and thereby their predictive performance.

To combat the prediction shift the authors introduced what they named *ordered boosting*, being a permutation-driven alternative to classic approaches. Ordered boosting was derived from a proposed *ordering principle*, inspired by online learning algorithms that sequentially receive training instances through time. The theoretical reasoning behind the ordering principle and ordered boosting is quite elaborate and complex, and we will therefore not

extensively cover it in this thesis, though the details can obviously be read about in the original paper [48]. Practically, however, ordered boosting operates by maintaining a set of models that were trained with separate data instances. These models are utilized when calculating the residual for a specific training instance by then using a model that was trained without this instance.

The result is that CatBoost's innovations successfully prevent the aforementioned prediction shift, and thereby achieve better prediction performance than earlier gradient boosting algorithms. However, this comes at the caveat of often requiring a large number of trained models, making CatBoost a sub-optimal choice for machine learning tasks with restrictions on model complexity and memory requirements.

2.2.6 Algorithm Comparisons

Generally, the methods of gradient boosting; XGBoost, LightGBM and CatBoost, are stronger than regular boosting methods, like AdaBoost, while each have their benefits and downsides. For instance, LightGBM is very efficient in terms of training time and memory consumption, but can sometimes struggle to achieve as good prediction performance as the other alternatives [31]. CatBoost achieves the strongest prediction performance, especially for datasets of catagorical data, but typically requires larger ensemble sizes than LightGBM and XGBoost to achieve this [48]. While XGBoost lands somewhere in between the other two methods. It is generally slower to train, but can achieve better prediction performance with smaller ensembles, based on its greedy and exhaustive tree-generation process [12, 31, 48].

2.3 Related Work

To our knowledge, attempting to improve upon gradient boosting methods by exploring per-tree hyperparameter configurations has not been researched in the past. The typical method of improving upon gradient boosting algorithms have mostly focused on altering the procedure of which trees are generated, and rarely on the structure of models produced. This can be demonstrated with the improvements made with XGBoost [12] from the original gradient boosting method [23], and those made thereafter with LightGBM [31] and CatBoost [48]. The closest examples to our topic, are a couple of papers that implemented gradient boosting where the diversity of ensemble trees were encouraged, which had positive findings [51, 5]. However, the promotion of diversity was here mostly centred around deterministically selecting the training data of each tree. Allowing diversity in the ensemble trees' hyperparameters was not investigated. In fact, hyperparameter related research regarding to gradient boosting almost exclusively focus on the optimization methods [49, 63], rather than obtaining insight or altering how the hyperparameters are handled in the algorithm. Some of the few exceptions are Probst et al. [47], who attempted to find optimal standard values of the hyperparameters of XGBoost, among other algorithms, and a quite recent paper by Ørebæk and Geitle [66], which explored XGBoost's hyperparmeters with the use of 3D visualizations.

However, that is not to say that having multiple contained hyperparameter configurations in an ensemble model is a completely novel concept. While this, to our knowledge, has not been investigated for gradient boosting ensembles, it is not uncommon in certain other types of ensembles. Most predominantly; *stacking* ensembles [62], and ensembles produced with

ensemble selection [11] or derivatives. These types of ensembles are different from gradient boosting ensembles in that the base learners do not feed into each other. They are individual models, of which individual predictions are in some way combined into a singular ensemble prediction. Both of these types of ensembles allow the base learners of a given ensemble to be full models produced by different machine learning methods, though they can also be homogeneous. As a result, it is natural that the base learners have separate hyperparameter configurations. There have been proposed several methods for how ensembles of these types should be built and optimized in relation to their hyperparameters. For instance Feurer et al. [18] proposed a *post hoc ensemble generation* procedure, based on ensemble selection. In this procedure, a regular hyperparameter optimization approach is carried out, typically with Bayesian Optimization, but the models produced with the explored configurations are stored and later greedily combined into an ensemble that maximizes prediction performance. Lévesque et al. [37] extended on this approach by combining the hyperparameter optimization procedure, based on Bayesian Optimization, with ensemble training. Here, each iteration of Bayesian Optimization optimizes a single base learner, and between interations, cycles through the base learners to optimize. Shahhosseini et al. [56] proposed a procedure of which base learners' hyperparameters and weights for prediction calculation were optimized in a nested process. Wistuba et al. [61] proposed a framework called Automatic Frankensteining, for automatically generating and tuning complex ensembles, here referred to as Frankenstein ensembles. And Wenzel et al. [60] designed "deep" ensemble approaches that generate ensembles, based on artificial neural networks, with focus on diversity in (neural network) weights and hyperparameters. All of these different approaches have shown great benefits of per-base learner hyperparameter configurations, and their optimization, in ensembles. Beyond this, there are several examples of these ensemble types being implemented in practical applications with positive results [45, 50, 52].

Another comparison to make with per-tree hyperparameters for gradient boosting ensembles is how artificial neural networks are hyperparameter optimized. Artificial neural networks consist of multiple layers, where each layer can be of a different type, have a different number of nodes, different activation functions, etc. The hyperparameters of neural networks are thus divided into two types: Global hyperparameters that apply to the whole network, of which the number of hidden layers is a good example, and per-layer hyperparameters defining only the characteristics their associated layer. Optimizing the per-layer hyperparameters in artificial neural networks is absolutely essential to achieving optimal prediction performance [16, 2]. Models of boosting ensembles and artificial neural networks are also relatively similar based on the fact that they are both "feed forwards" oriented, meaning the inputs are received, transformed and passed from one "component" to another.

Chapter 3

Flexible Ensemble Structure Optimization

The research questions explore per-tree hyperparameter configurations with gradient boosting ensembles. In this chapter, we define this concept as what we call "flexible ensemble structures". To explain the concept, in Section 3.1, we first outline how boosting ensembles are traditionally structured with illustrations and descriptions, and compare these to that of flexible ensemble structures. Thereafter, in Section 3.2, we define and discuss different possible approaches to how flexible ensemble structures can be optimized. Specifically, we define two general approaches, Holistic and Incremental flexible ensemble structure optimization, demonstrated with illustrations and pseudo codes.

3.1 Flexible Ensemble Structures

As discussed in Section 2.1, hyperparameters influence the training procedure and/or the architecture of produced machine learning models. In the context of gradient boosting ensembles, instances of architecture can be exemplified as the number of trees (the ensemble size) or the trees' depth. For this thesis, we define a more general term, *structure*, which refers to any and all effects of hyperparameters on produced ensembles. Gradient boosting ensemble structures are traditionally defined by only one set of hyperparameters. In practise, this means that all ensemble trees are uniformly defined by the same hyperparameter configuration. An example of a traditional ensemble structure of 3 trees is illustarated in Figure 3.1 However, we theorize that traditional structures, with their uniformly configured trees, are preventing gradient boosting ensembles from achieving their full predictive potential. Hyperparameter values are already essential to achieving the best possible prediction performance on machine learning tasks and need to be custom tuned to each dataset. In the gradient boosting ensemble building process, each tree is generated to minimize the ensemble's prediction errors, based on where the previous tree left off. In practise, this means that each tree is given a slightly different prediction problem, and thereby a slightly different dataset. It therefore seems reasonable to theorize that optimizing each tree for their respective prediction problem, by giving them individual hyperparameter configurations, would be more optimal for prediction performance.

We call the concept of having individual hyperparameter configurations per ensemble tree, *flexible ensemble structures*, of which an example is illustrated in Figure 3.2. Compared to traditional structures, we can see that all trees in this example ensemble have individual

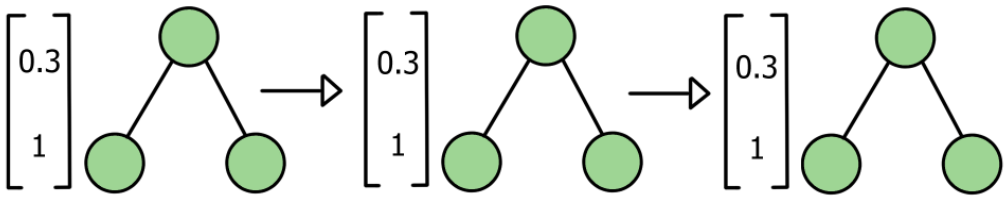


Figure 3.1: Visual example of a traditional ensemble structure of 3 trees. The trees are symbolised as line-connected collections of nodes, each with an associated hyperparameter configuration, in cartesian coordinate format, on their left. The arrows indicate how the trees feed into each other. Moving left to right, the left furthermost ensemble tree is thereby the first, and the furthermost right, the last. In regards to the hyperparmeter configurations, these are illustrated to contain two parameters, one continuous and one integer, and are identical across all three trees. This demonstrates the very limited flexibility between the trees' regularization, and thereby their individual predictive capabilities.

hyperparameter configurations. In practice, this means that each tree can be completely different in all aspects, from their size to how they are generated. Of course, this level of flexibility also allows for the trees to be identical in structure, should this be optimal.

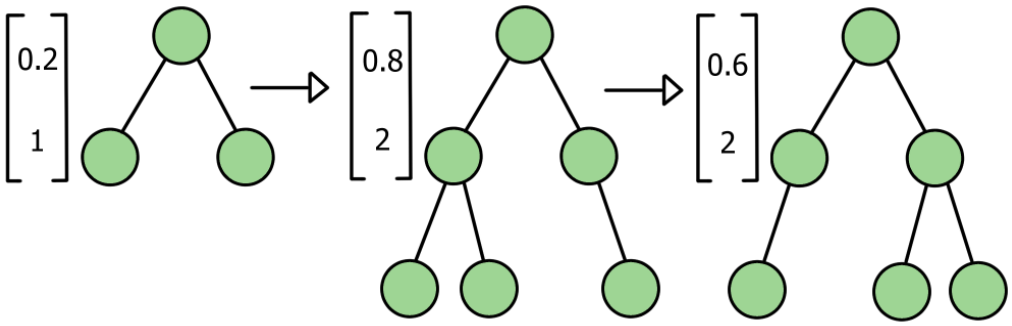


Figure 3.2: Visual example of a flexible ensemble structure of 3 decision trees based on individual hyperparmeter configurations. The trees are symbolised as line-connected collections of nodes, each with an associated hyperparameter configuration, in cartesian coordinate format, on their left. The arrows indicate how the trees feed into each other. Moving left to right, the left furthermost ensemble tree is thereby the first, and the furthermost right, the last. In regards to the hyperparmeter configurations, these are illustrated to contain two parameters, one continuous and one integer, and are individual for each tree. This demonstrates the increased flexibility between the trees' regularization, and thereby their individual predictive capabilities, compared to traditional structures.

3.2 Approaches to Flexible Ensemble Structure Optimization

To explore the research question we needed to establish a few different approaches of flexible ensemble structure optimization. Specifically, we explored two approaches; The first was to optimize all per-tree hyperparameter configurations in the same procedure. We call this, *Holistic flexible ensemble structure optimization*. The second was to optimize each

tree individually and separately from from the others, specifically in tandem with their generation in the ensemble building process. We call this, *Incremental flexible ensemble structure optimization*.

In the sections below, we discuss the two optimization approaches to clearly demonstrate their building and optimization processes, as well as outline their pseudo code. The pseudo codes are made neutral to implementation details, such as how models are trained and evaluated. Instead, they contain an associated objective-function, based on input-influenced output values, to be optimized with a selected hyperparameter optimization method. This objective-function is further referred to as the *Objective*.

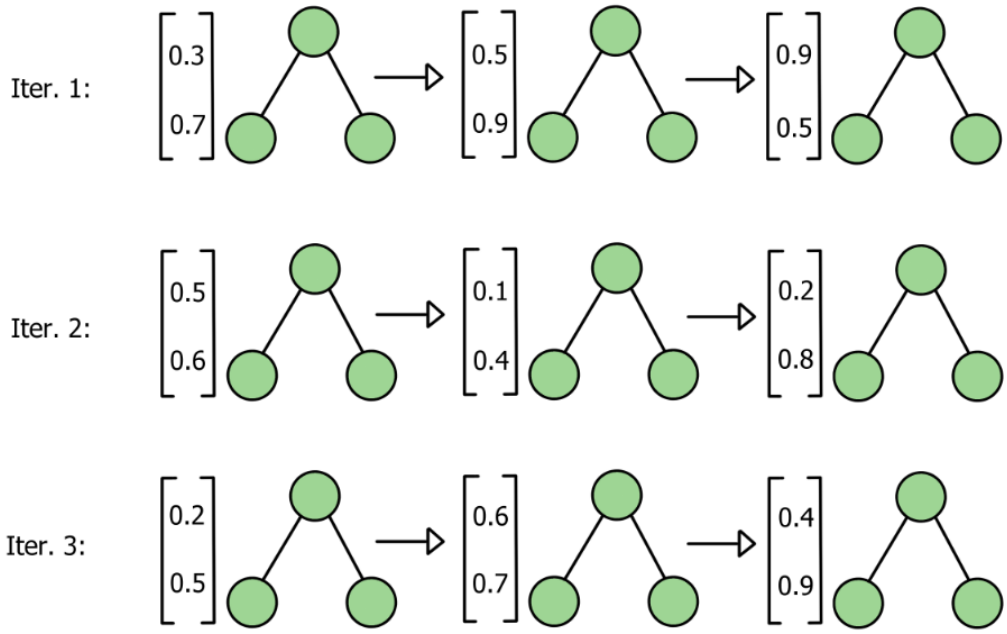


Figure 3.3: Visual demonstration of Holistic optimization of flexible ensemble structures. Ensembles are represented as collections of nodes (trees), with associated cartesian coordinate hyperparameter confugurations on their left, connected by arrows indicating input/output flow. The visualization contains three ensemble instances, each representing the result of one search iteration with Holistic flexible ensemble structure optimization. Comparing these iterations, we can see that each one generates a full-sized ensemble with individual hyperparameter configurations for each contained tree.

3.2.1 Holistic Flexible Ensemble Structure Optimization

Holistic flexible ensemble structure optimization is inherently quite similar to how traditional structures are optimized, where all hyperparameters are handled simultaneously. This is of course under the presumption that a pipeline optimization-process is not used. However, where traditional structures only have one configuration containing a certain number of hyperparameters, flexible structures have one such configuration for each tree in the ensemble. The benefit of this approach is that produced ensembles, with their contained per-tree hyperparameter configurations, are evaluated as a whole. This means that each tree's configuration will be optimized to aid the entire ensemble's prediction performance.

The downside of optimizing flexible structures holistically, is that this practically means that the number of hyperparameters to be optimized simultaneously increases linearly with the ensemble size. This is a potential problem because the search complexity, in terms of unique value combinations, increases exponentially for each added hyperparameter [64]. This optimization problem thus becomes comparable to that of neural networks. Neural networks can have many layers that can be of different types, contain different numbers of nodes, use different activation functions etc. Optimizing the structure of neural networks is therefore a task that gets exponentially more complex the larger the size. Regardless, due to the benefits of large (deep) neural networks, optimization strategies are continually being developed and improved upon [16, 65, 43]. This demonstrates that even though investigating Holistic flexible ensemble structure optimization can seem like a daunting task, especially with larger ensembles, it may yet be possible and worth exploring for its benefits. Holistic flexible ensemble structure optimization is visually demonstrated in Figure 3.3.

3.2.1.1 Procedure

The pseudo code of Holistic optimization is contained in Figure 3.4. The procedure has three input parameters. The first, C, is a positive integer that denotes the number of trees the produced ensemble is to contain. The second, I, is a positive integer that denotes number of iterations of a selected hyperparameter optimizaton method the ensemble is to be optimized through. And the third, S, denotes the configuration search space, being a set of value-ranges, each based on a lower value, l, and an upper value, u. Note that each range is here associated with a unique hyperparameter, of which the total number is denoted as M. The value type of each range is therefore dependent on the associated hyperparameter.

```
procedure HOLISTIC(C, I ∈ ℕ, S = {range of (l_1, u_1), ..., range of (l_M, u_M)})
    procedure OBJECTIVE(H = flexible ensemble structure configuration
                            represented as an array of C per-tree
                            hyperparameter configurations in the
                            form (h_1, ..., h_M), where each h is a
                            value for a unique hyperparameter)
        E_0 ← empty ensemble
        for c ← 1 to C do
            t_c ← boosting tree with H_c per-tree hyperparameter configuration
            E_c ← t_c added to E_{c−1}
        end for
        p ← prediction performance of E_C
        return p, E_C
    end procedure
    E_best ← the best ensemble of I iterations of a hyperparameter
    optimization method, each producing a flexible ensemble structure
    configuration, H, based on the search space, S, evaluated on the
    Objective
    return E_best
end procedure
```

Figure 3.4: Pseudo code for Holistic flexible ensemble structure optimization.

Most of the Holistic flexible ensemble structure optimization process is contained within the *Objective*. The Objective has one input parameter passed from the optimization method; H, being a flexible ensemble structure configuration. The structure is represented as an array containing C per-tree hyperparameter configurations, each in the form (h_1, ..., h_M), where each h is a value for a unique hyperparameter. The Objective begins by initializing an empty ensemble, E_0. This ensemble is then gradually built one tree at a time, as denoted by the for-loop with a total of C run-troughs, For each run-through, c, a boosting tree t_c is generated based on its associated per-tree hyperparameter configuration in H, H_c, before it is added to the ensemble, denoted as E_c. Once the ensemble is finalized, denoted as E_C, its prediction performance, p, is measured. Finally, p and E_C are returned as the result of the Objective. The Objective is optimized through I iterations of the optimization method, each producing a flexible ensemble structure configuration, H, based on the search space, S, and evaluating it on the Objective. After all iterations, the best performing ensemble, E_{best}, is returned as the result of Holistic flexible ensemble structure optimization.

3.2.2 Incremental Flexible Ensemble Structure Optimization

Contrary to the Holistic approach, Incremental flexible ensemble structure optimization is handled quite differently from traditional structures. For this approach, the ensemble trees are not optimized as a unit, but instead, each in their own isolated procedure. While this approach can be handled in various ways, we chose to optimize the trees in tandem with their generation in the training process. This way, the hyperparameter optimization task can be treated like a gradient decent problem, and can be conceptualized as an extension of how gradient boosting is already designed, as outlined in Section 2.2.2. This would also largely abstract the hyperparameters away from users, which would make applications of boosting algorithms significantly simpler. Search difficulty would also be much more manageable compared to the Holistic approach, only optimizing the hyperparameters of one tree at at a time. However, the potential downside of this method is that each tree is optimized as an extension of the current ensemble. This means that each tree is optimized without the consideration of proceeding trees, which could negatively impact the predictive abilities of finalized ensembles. Incremental flexible ensemble structure optimization is visually demonstrated in Figure 3.5.

3.2.2.1 Procedure

The pseudo code for Incremental flexible ensemble structure optimization is contained in Figure 3.6. Just as with the Holistic approach, the procedure has three input parameters. The first; C, denoting the number of trees the produced ensemble is to contain. The second; I, denoting number of iterations of a selected hyperparameter optimization method each tree is to be optimized through. And the third; S, denoting the configuration search space, being a set of M unique hyperparameter value-ranges, each based on a lower value, l, and an upper value, u.

Incremental flexible ensemble structure optimization begins by initializing an empty ensemble, E_0. As with the Holistic approach, this ensemble is gradually built, tree by tree, through a for-loop with a total of C run-throughs. However, differently from the Holistic approach, where this process is contained within the Objective, the opposite is the case for Incremental flexible ensemble structure optimization. The Objective has one input-parameter; H, being one per-tree hyperparameter configuration in the form (h_1,

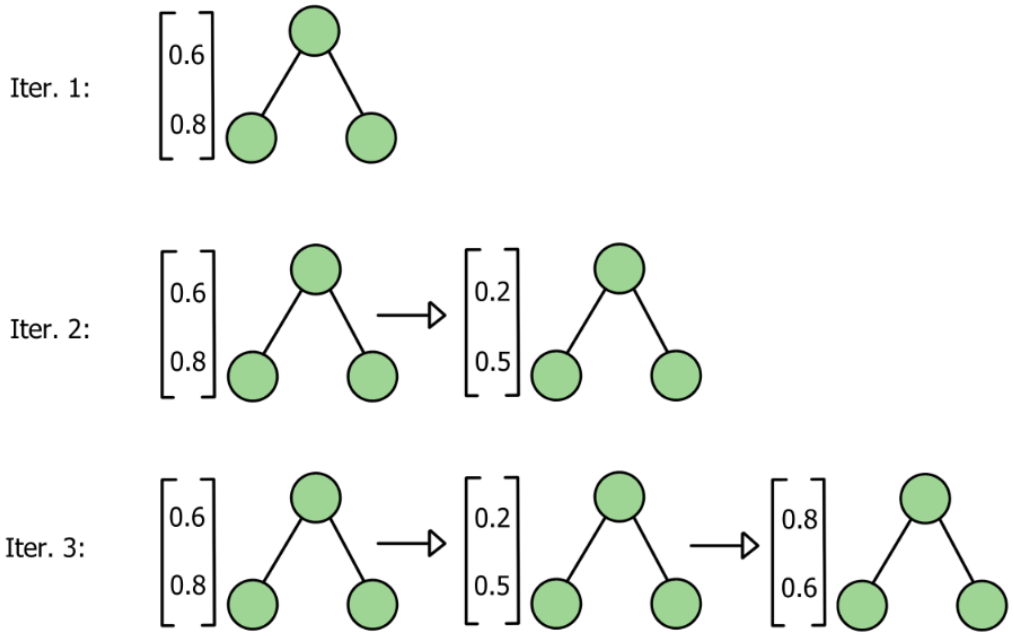


Figure 3.5: Visual demonstration of Incremental optimization of flexible ensemble structures. Ensembles are represented as as collections of nodes (trees), with associated cartesian coordinate hyperparameter configurations on their left, connected by arrows indicating input/output flow. The visualization contains three ensemble instances, each representing the result of one iteration of Incremental flexible ensemble structure optimization. Comparing these iterations, we can see that each iteration hyperparameter optimizes and adds one tree to the ensemble.

procedure INCREMENTAL(C, $I \in \mathbb{N}$, $S = \{$range of (l_1, u_1), ..., range of $(l_M, u_M)\}$)
  $E_0 \leftarrow$ empty ensemble
  for $c \leftarrow 1$ **to** C **do**
    procedure OBJECTIVE(H = per-tree hyperparameter configuration in the form $(h_1, ..., h_M)$, where each h is a value for a unique hyperparameter)
      $t \leftarrow$ boosting tree with H per-tree hyperparameter configuration
      $E_{ext} \leftarrow t$ added to E_{c-1}
      $p_{ext} \leftarrow$ prediction performance of E_{ext}
      return p_{ext}, E_{ext}
    end procedure
    $E_c \leftarrow$ best extended ensemble of I iterations of a hyperparameter optimization method, each producing a per-tree hyperparameter configuration, H, based on the search space, S, evaluated on the Objective
  end for
  return E_C
end procedure

Figure 3.6: Pseudo code for Incremental flexible ensemble structure optimization.

..., h_M), where each h is a value for a unique hyperparameter. The Objective begins by defining a boosting tree, t, with its associated hyperparameter configuration, H. The current ensemble, E_{c-1}, is then extended with t, denoted as E_{ext}, and its prediction performance, p_{ext}, is measured, before both p_{ext} and E_{ext} are returned as the result of of the Objective. For each run-though, c, of the for-loop, the selected hyperparameter optimization method is ran with I iterations, each producing a per-tree hyperparameter configuration, H, based on the search space, S, and evaluating it on the Objective. After all iterations, the extended ensemble of best performance is returned and defined as the updated ensemble, E_c. After all trees are added, the finalized ensemble, E_C, is returned as the result of Incremental flexible ensemble structure optimization.

Chapter 4

Groundwork for the Experiments

To answer the research questions, we conducted 5 experiments, of which methodologies are described in Chapter 5 and results are presented in Chapter 6. This chapter outlines the groundwork for these experiments. Section 4.1, outlines the different datasets and how they were selected, preprocessed and baselined. Section 4.2 discusses the selection of the algorithm used in experiments, XGBoost. In Section 4.3 the selected hyperparameters of XGBooost are discussed. Section 4.4 discusses the hyperparmeter optimization methods used in the experiments. Finally, Section 4.5 outlines the hardware used for computations and how this influenced the experiments.

4.1 Datasets

To answer the research questions, we needed to select a few different datasets to serve as optimization tasks. To promote generalizable results, we selected both regression and classification datasets of varying characteristics, such as size, number of features, feature datatypes, range of targets, and skewness in target distributions. To ensure fast runtimes during investigations, we limited the size of datasets to be no larger than 20000 datainstances. The necessity for this is discussed in Section 4.5. An overview of the selected regression datasets are tabulated in Table 4.1, while the selected classification datasets are tabulated in Table 4.2. These datasets were used in all experiments.

Dataset	Instances	Nr. Features	Feature Type(s)	Value Range
Concrete[1]	1030	8	Real	2.33 - 82
Energy Prediction[2]	19735	27	Real	10 - 1080
Housing[3]	506	13	Real	5 -50
Seoul Bike Sharing[4]	8760	13 (17)	Real & Categorical	0 - 3418

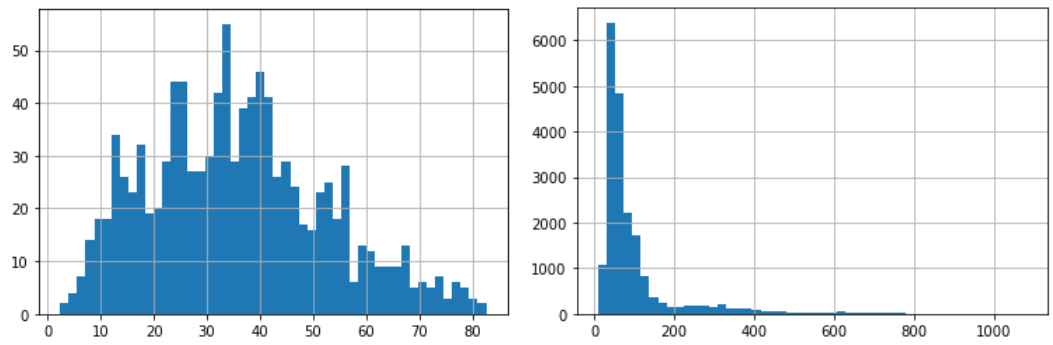
Table 4.1: The selected regression datasets used to train and evaluate ensembles.

[1] https://archive.ics.uci.edu/ml/datasets/Concrete+Compressive+Strength
[2] http://archive.ics.uci.edu/ml/datasets/Appliances+energy+prediction
[3] https://archive.ics.uci.edu/ml/machine-learning-databases/housing/?C=N;O=D
[4] https://archive.ics.uci.edu/ml/datasets/Seoul+Bike+Sharing+Demand
[5] https://archive.ics.uci.edu/ml/datasets/car+evaluation
[6] http://archive.ics.uci.edu/ml/datasets/Statlog+(Landsat+Satellite)
[7] https://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/

Dataset	Instances	Nr. Features	Feature Type(s)	N Classes
Car Evaluation[5]	1728	6	Ordinal	4
Statlog Satelite[6]	6435	36	Real	6
Winequality-red[7]	1599	11	Real	6

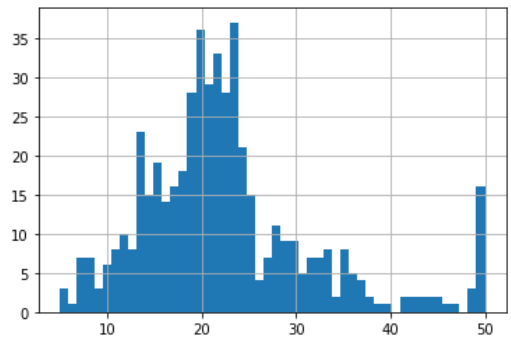
Table 4.2: The selected classification datasets used to train and evaluate ensembles.

To investigate the datasets' skewness of outputs, we generated histograms with 50 bins for regression, and bins equal to the number of classes for classification. Of the regression datasets, illustrated in Figure 4.1, we can see that Concrete (Subfigure (a)) and Housing (Subfigure (c)) are relatively balanced with what seems like normal distributions of outputs. Energy Prediction (Subfigure (b)) and SeoulBike Sharing (Subfigure (d)), on the other hand, are both quite skewed towards the lower end of the output value spectrum. Specifically, while Energy Predictions outputs range from 0 to 1080, most of the outputs are located within 0 to 200, and for Seoul Bike Sharing, most outputs are located within 0 to 1500 out of the total 0 to 3418 value range.

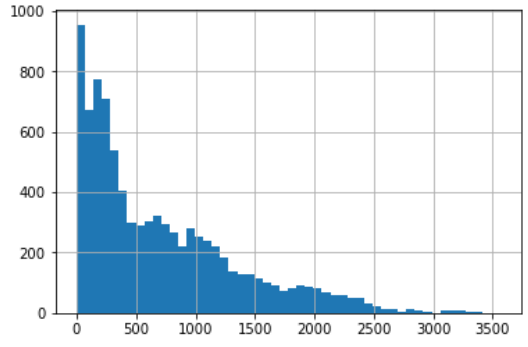


(a) Concrete

(b) Energy Prediction



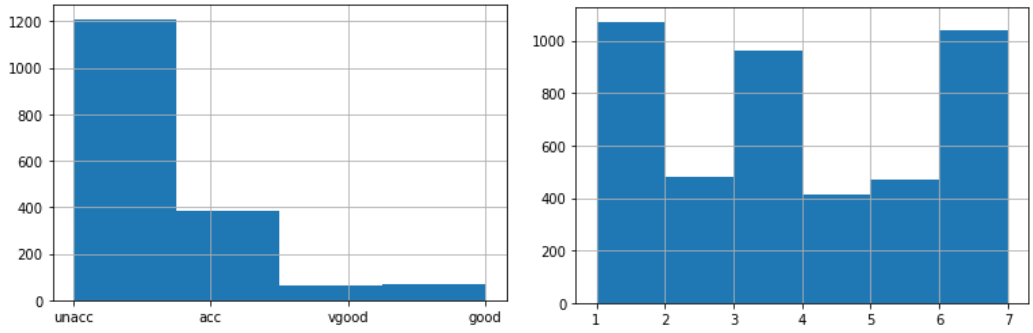
(c) Housing



(d) Seoul Bike Sharing

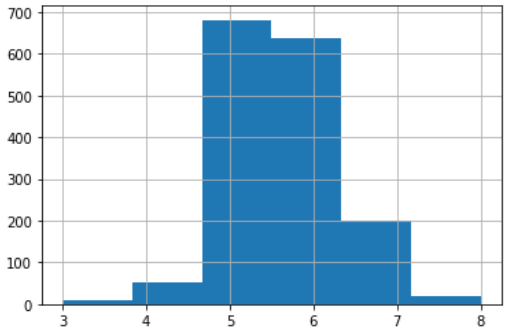
Figure 4.1: Histograms demonstrating the skewness of the regression datasets's outputs.

Of the classification datasets, illustrated in Figure 4.2, Car Evaluation (Subfigure (a)) and Winequality Red (Subfigure (c)) are similarly skewed towards half of their respective classes, while Statlog Satelite (Subfigure (b)) is more evenly distributed in its classes.



(a) Car Evaluation

(b) Statlog Satelite



(c) Winequality Red

Figure 4.2: Histograms demonstrating the skewness of the classification dataset's outputs

4.1.1 Preprocessing

As per common machine learning practice, the selected datasets were split into training-sets used to generate models, and test-sets to evaluate their performance. To ensure that outputs were equally represented in both the training and test setting, all datasets were split with stratified random sampling at a ratio of 80% for training and 20% for testing. An exception to this treatment was, however, given to Statlog Satelite which had predetermined training- and test-sets with proportionally similar output distributions. While this treatment was applied for all datasets, these splits were not used for most experiments. The reasoning for this is that cross validation is typically more fit for hyperparameter optimization tasks, and as most of the datasets were relatively small in size, we prioritized a larger number of training data by running cross validation on the full datasets. For datasets containing categorical text-values, these were converted to numerical values through one-hot encoding, as this is a good way of representing the nature of categorical values to machine learning algorithms [9]. Data-instances with missing values were kept, while any id- or date-columns were removed from the datasets, as these do not provide any benefits to the learning process.

4.1.2 Baselines

An additional factor we needed to take into account in the datasets' selection was gradient boosting algorithms being the used machine learning method. Gradient boosting algorithms are known for their ability to solve complex, large-scale problems with minimal resources

[12]. Because of this, a potential occurence when working with smaller datasets is perfect prediction rates, even at low numbers of ensemble trees. This would be problematic, as distinguishing between the effectiveness of investigated approaches to optimizing flexible ensemble structures would be impossible. We therefore needed to ensure that selected datasets were challenging enough to not be perfectly predicted by a given gradient boosting algorithm at small ensemble sizes. We therefore baselined all selected datasets on XGBoost, LightGBM and CatBoost with various hyperparmeter configurations, and controlled them to not have a 100% correct test set prediction rate.

Datasets	Algorithm	Standard Parameters		n_estimators=1 learning_rate=0.5		n_estimators=10 learning_rate=0.5	
		Train	Test	Train	Test	Train	Test
Concrete	XGBoost	**0.40**	3.07	18.13	18.80	**1.85**	4.15
	LightGBM	1.40	3.07	**8.45**	**9.16**	2.70	**4.05**
	CatBoost	1.27	**2.71**	9.75	9.58	3.75	4.61
Energy Prediction	XGBoost	**22.82**	**37.01**	**52.58**	**51.77**	**38.04**	**43.78**
	LightGBM	32.34	38.75	54.31	54.53	41.75	45.14
	CatBoost	30.00	37.11	56.59	55.88	47.47	48.01
Housing	XGBoost	**0.01**	1.95	11.37	10.91	0.89	2.30
	LightGBM	0.91	2.20	**4.22**	**4.02**	**1.63**	**2.16**
	CatBoost	0.44	**1.75**	4.62	4.24	1.98	2.45
Seoul Bike Sharing	XGBoost	**71.27**	148.	371.75	379.05	**127.35**	**154.39**
	LightGBM	108.88	**140.60**	**320.83**	**327.69**	140.08	159.98
	CatBoost	104.91	142.89	350.19	353.66	168.67	176.74

Table 4.3: The Baseline for the regression datasets on XGBoost, LightGBM and CatBoost. The ensembles were trained on the training-set and evaluated on the test-set of each dataset. MAE was used as the performance metric. Bold values indicate the best prediction performance for a given hyperparameter configuration.

Datasets	Algorithm	Standard Parameters		n_estimators=1 learning_rate=0.5		n_estimators=10 learning_rate=0.5	
		Train	Test	Train	Test	Train	Test
Car Evaluation	XGBoost	**100%**	**97.97%**	**95.51%**	**93.06%**	99.78%	**97.97%**
	LightGBM	94.57%	92.48%	94.57%	92.48%	**100%**	**97.97%**
	CatBoost	**100%**	97.39%	81.11%	81.21%	98.76%	94.79%
Statlog Satellite	XGBoost	**100%**	90.15%	**92.26%**	**85.80%**	98.12%	**88.85%**
	LightGBM	**100%**	90.70%	89.44%	84.55%	**99.70%**	**88.85%**
	CatBoost	99.93%	**91.05%**	79.41%	76.50%	89.22%	85.00%
Winequality-red	XGBoost	**100%**	67.50%	**74.43%**	**61.56%**	**93.66%**	**64.37%**
	LightGBM	**100%**	**67.81%**	65.83%	58.12%	88.89%	62.50%
	CatBoost	**100%**	66.87%	58.32%	56.25%	68.56%	58.43%

Table 4.4: The Baseline for the classification datasets on XGBoost, LightGBM and CatBoost. The ensembles were trained on the training-set and evaluated on the test-set of each dataset. Accuracy was used as the performance metric. Bold values indicate the best prediction performance for a given hyperparameter configuration.

The baselines for the standard parameters, and two more tested configurations are tabulated in Table 4.3 (regression) and Table 4.4 (classification). The train/test splits are described in Section 4.1.1.

Naturally, we also needed to ensure that the datasets' outputs were possible to predict at rates higher than random guessing, as the alternative would practically result in the same problem as with perfect prediction rates. We therefore additionally controlled the classification datasets to have a Cohen's Kappa Score greater than 0, while for regression we compared the prediction performance with different ensemble sizes (values of n_estimators) to control that prediction performance improvement was possible.

4.2 Selected Gradient Boosting Algorithm: XGBoost

With the research questions being centered around the effects of optimizing flexible ensemble structures with gradient boosting algorithms, we needed to establish a gradient boosting algorithm to use in experiments. We were thereby left to decide between two choices; designing and implementing a boosting algorithm from scratch, or use a preexisting boosting algorithm implementation to build and test upon. We chose the latter, as implementing a boosting algorithm from scratch would likely be a cumbersome and extensive process which could result in bugs and problems influencing the validity of experiment results. Such a ground-up implementation would most likely also be inefficient in terms of computation time. This is undesirable, as hyperparameter investigations tend to be computationally expensive.

We chose XGBoost as the algorithm to use in experiments. This decision was based on the discussions in Section 2.2.6, and the baselines in Section 4.1.2, where XGBoost and LightGBM were competitive, but XGBoost seemed to come out slightly ahead. XGBoost also has good documentation[8] and the simplest logic of the discussed gradient boosting algorithms. This makes potential modifications of the source code more approachable. Relevant to this, we needed to generate ensembles with per-tree hyperparameter configurations, something that, to our knowledge, is not by standard supported with XGBoost. To accomplish this, we instead used XGBoost's method of saving and continuing training processes[9]. In relation to this method, we found that upon loading a saved model, the hyperparameters of the previous training procedure were kept up until the point of continuation. We could thereby iteratively build flexible ensemble structures through the following procedure:

1. Define the hyperparameters for the tree to be added.
2. Load the current model. (Skip if first tree)
3. Train one tree. (n_estimators = 1)
4. Save model.
5. Repeat 1 to 4 until ensemble is complete.

4.3 Selected Hyperparameters

As experiments largely revolved around hyperparameter optimization, we needed to select a set of XGBoost's hyperparameters to optimize in experiments. For this purpose, we selected learning_rate, max_depth, subsample and colsample_bytree.

[8] `https://xgboost.readthedocs.io/en/latest/python/python_intro.html`
[9] `https://xgboost.readthedocs.io/en/latest/tutorials/saving_model.html`

Learning_rate is a parameter that scales the tree's influence on ensemble predictions as a whole. The motivation behind selecting this parameter was the possibility that the trees have different levels of "value" to ensemble prediction, in which case, individual levels of influence for the different trees seems likely to be beneficial. Values of learning_rate are continuous and typically range between 0 and 1, which is also the search range we used in experiments.

Max_depth is a parameter that regulates the size of trees, with the purpose of explicitly preventing overfitting by restricting trees' ability to accurately model their training data, thereby forcing them to make more general predictions. It is theoretically possible that ensemble trees should optimally be diverse in generalizability, depending on their individual prediction problems. This was in turn the motivation behind the selection of this parameter. Max_depth values are integers that can technically be any whole number equal to or above 1. However, as the trees' natural maximum depth is partially determined by the amount of dataset features, we decided to use 1 to the number of features as the search range for this parameter.

A similar motivation to that of max_depth was relevant to the selection of subsample and colsample_bytree, which are also regularization parameters, though comparably more implicit in their effects. Subsampling is the principle of generating each decision tree from a randomly selected subset of the available training-data. This implicitly enforces ensemble generalizability by encouraging diversity between trees. The subsample parameter specifically controls the size of the subsampled dataset based on a percentage of the full dataset size. Column sampling is principally similar to subsampling, but is instead applied to dataset features (columns). Colsample_bytree in particular, controls the number of features to be randomly selected for tree-generation, based on a percentage value of the total number of features. Due to representing percentages, values of both subsample and colsample_bytree are continuous, and between 0 and 1. However, due to the fact that training on too few data instances or features can counter-productively lead to poor prediction performance, we use 0.6 to 1 as the search range for subsample, and 0.8 to 1 for colsample_bytree.

Another important hyperparameter of XGBoost is n_estimators, which controls the number of ensemble trees. The number of trees is important to ensembles' prediction performance because it directly influences the number of iterations errors are minimized. However, because the research questions explore the effects of flexible ensemble structures, it is implicit that they needed to be compared with traditional structures. Optimizing n_estimators dynamically, would therefore complicate things, as it would become difficult to distinguish between differences based on n_estimators, and ones based on the other hyperparameters. Therefore, we instead used a static n_estimators value for all experiments. Because the optimization difficulty of flexible ensemble structures is unknown, we set this value as 5. This way ensembles would be large enough for us to observe observe potential effects based on flexible ensemble structures, while (hopefully) small enough to keep search difficulty manageable.

Note that while these are the selected hyperparameters for the experiments, not all of the experiments optimize every hyperparameter. In cases where hyperparameters are left out of optimization processes, we used the standard parameters for these, except for n_estimators, where 5 was used regardless. The standard values for the selected hyperparameters and their used search ranges are tabulated in Table 4.5.

Hyperparameter	Standard Parameter	Used Search Range
learning_rate	0.3	0.01 - 1
max_depth	6	1 - N. Features
subsample	1.0	0.6 - 1
colsample_bytree	1.0	0.8 - 1
n_estimators	100 (5 is used)	5

Table 4.5: The standard hyperparameter values of XGBoost, and search ranges used in experiments.

4.4 Selected Hyperparameter Optimization Methods

Different hyperparameter optimization methods have different strengths and weaknesses. The conducted experiments were varied in objective, but can be abstracted to primarily obtain insights based on one of two aspects; Configurations of good prediction performance and more general aspects of configuration landscapes.

As discussed in greater detail in Section 2.1.5, Bayesian Optimization is one of the most efficient hyperparameter optimization methods, in regards to number of iterations, due to its ability to dynamically adjust its search procedure based on earlier evaluations. We therefore used this method in experiments where obtaining well performing configurations was the primary necessity. Specifically, we used the Hyperopt[10] implementation.

Observing aspects related to configuration landscapes requires observations of a wide variety of configurations. Random Search was a good choice of method for this purpose. It generates completely random configurations, and generally searches quite evenly within each hyperparameter value range, given enough iterations [3]. Random Search was therefore used in experiments where insights based on configuration landscapes were the focal point.

4.5 Hardware and Computational Resources.

The experiments in the following chapters were ran on a computer with an Intel i5-4670K CPU, 16 GB of RAM, and an Nvidia GTX 970 graphics card. However, due to implementation details, the graphics card could not be used in computations. These relied solely on the CPU and available RAM. Due to the relatively outdated CPU, aspects such as the number of search iterations in optimization processes and dataset sizes, needed to be kept relatively low to run the experiments in a reasonable amount of time. This need was further reinforced by the fact that the implementation details of flexible ensemble structures with XGBoost required reading and writing to disk for each tree in the building process, as discussed in Section 4.2.

[10] `http://hyperopt.github.io/hyperopt/`

Chapter 5

Experiments

To the extent of our knowledge, flexible ensemble structures have not yet been explored with XGBoost or other gradient boosting algorithms. As a result we do not know to what extent flexible structures are useful. The first and most necessary aspect for evaluating flexible ensembles structures usefulness, is establishing whether there is a significant benefit to prediction performance by utilizing flexible ensemble structures (RQ1), compared to traditional ones. A second aspect, quite necessary for establishing the applicability of flexible ensemble structures, is how difficult they are to optimize (RQ2). This is necessary due to the fact that the number of unique flexible ensemble structure configurations increases exponentially with more trees. This creates a possibility that that the difficulty of finding optimally performing configurations could increase similarly, which would increase requirements for computational resources, and thereby discourage the application of flexible structures. And considering the lack of research into flexible ensemble structures with gradient boosting algorithms, we also needed to investigate how flexible ensemble structures can and should be optimized (RQ3).

This chapter outlines the methodology of the five experiments conducted in this thesis. Specifically, the methodology of Experiment 1 through 5 are contained in Section 5.1 through 5.5, respectively. The experiment results are presented in Chapter 6.

5.1 Experiment 1

Experiment 1 was conducted to obtain insight into RQ1 by comparing the prediction performance of flexible and traditional ensemble structures. To achieve this, we needed to obtain as optimal prediction performance as possible for each structure type and compare the values to get a measure of how much better or worse flexible structures are in this respect. Therefore, we implemented Bayesian Optimization as the method of optimizing configurations for both traditional and flexible ensemble structures. The optimized ensembles to be compared consisted of 5 trees, and their structure configurations were evaluated with 2 repetitions of 5-fold cross validation. Mean Absolute Error was used as the prediction performance metric for the regression datasets, while Error was used for the classification datasets. For the traditional structures, Bayesian Optimization was ran with 1000 iterations, which we observed to be more than enough to converge prediction performance for this structure type. The flexible ensemble structures were optimized through 2000 evaluations to compensate for the difference in search complexity and ensure that the obtained structure configurations were sufficiently optimized.

While looking at raw difference in prediction performance values can give a give a certain amount of insight into which structure type is superior, the significance of the differences are not obvious from these values alone. Thus, to determine a significance of difference, the prediction performance values of the traditional and flexible ensemble structures needed to be compared relative to something else. For this reason, we additionally measured the difference between a given 5-tree traditional ensemble structure to one of 6 trees (extended by one tree), separately optimized. We could then obtain a measure of significance by comparing the differences between the flexible and traditional structured ensembles of 5 trees, to the differences between the traditionally structured ensembles of 5 and 6 trees. Specifically, we used the prediction performance of the 5-tree traditional ensemble structure as the baseline value, and calculated the percentage of improvement obtained with the flexible ensemble structure, relative to the improvement obtained by extending the traditional structure by one tree. We further refer to this as the *percentage of improvement* in prediction performance with a given flexible ensemble structure.

The comparison between the prediction performance of traditional and flexible ensemble structures was carried out on all datasets outlined in Section 4.1. For each dataset, we investigated four scenarios of hyperparameter optimization. Meaning combinations of the hyperparameters outlined in section 4.3. These scenarios were:

Scenario 1: learning_rate optimized in isolation.

Scenario 2: max_depth optimized in isolation.

Scenario 3: learning_rate, max_depth and subsample optimized in isolation.

Scenario 4: learning_rate, max_depth, subsample and colsample_bytree optimized in isolation.

5.1.1 Flexible Ensemble Structure Procedure

The pseudo code for the flexible ensemble structure generation procedure of Experiment 1 is contained in Figure 5.1, and builds upon the pseudo code of the Holistic approach, as outlined in Section 3.2.1. Of the procedure's input-parameters; C, I and S remain the same, respectively denoting the number of ensemble trees, the number of optimization method iterations, and the search space. X is an added input-parameter denoting the dataset, being a set of N data instances, each in the form $(\mathbf{x}, y)$.

The specific inputs used were as follows: X was set to the full version of the relevant dataset, C was set to 5, I was set to 2000, while S was set to be a combination of ranges for learning_rate, max_depth, subsample and colsample_bytree, depending on the relevant hyperparameter scenario. The ranges were the same as outlined in Section 4.3; For learning_rate, it was 0 to 1, with values being uniformly selected decimal numbers; for max_depth, 1 to the number of features in the relevant dataset, with values being uniformly selected integers; the range for subsample was 0.6 to 1, with values being uniformly selected decimal numbers; and the range for subsample, was 0.8 to 1, with values also being uniformly selected decimal numbers.

As the flexible ensemble structure generation procedure of Experiment 1 is based on Holistic flexible ensemble structure optimization, the overall pseudo code is quite similar to that outlined in Figure 3.4 of Section 3.2.1. However, it differs in returned objects and evaluation method, being cross validation. The Objective's input-parameter, H, is an array

```
procedure EXPERIMENT_1(X = {(x_1, y_1), ..., (x_N, y_N)},
                       C, I ∈ ℕ, S = {range of (l_1, u_1),
                       ..., range of (l_M, u_M)})
    procedure OBJECTIVE(H = flexible ensemble structure configuration
                        represented as an array of C per-tree
                        hyperparameter configurations in the
                        form (h_1, ..., h_M), where each h is a
                        value for a unique hyperparameter)
        P ← ∅
        F ← set of cross validation folds in the form (X_train, X_val), where
        each X_train and X_val are subsets of X
        for each X_train, X_val in F do
            E_0 ← empty ensemble
            for c ← 1 to C do
                t_c ← XGBoost tree with H_c per-tree hyperparameter configuration
                E_c ← t_c added to E_{c−1}
            end for
            p ← prediction performance of E_C trained on X_train and evaluated on
            X_val
            P ← P ∪ p
        end for
        return average(P)
    end procedure
    p_best, H_best ← best prediction performance and ensemble configuration
    of I iterations of Bayesian Optimization, each producing a flexible
    ensemble structure configuration, H, based on the search space, S,
    evaluated on the Objective
    return p_best, H_best
end procedure
```

Figure 5.1: The pseudo code for the generation procedure of one flexible ensemble structure in Experiment 1. The procedure is based on the Holistic approach to flexible ensemble structure optimization.

representing a flexible ensemble structure configuration. The Objective begins by initializing an empty set, P, to contain the prediction performance evaluated on the different cross validation folds. The training- and validation-folds are initialized in the set F, each in the form (X_{train}, X_{val}), where X_{train} and X_{val} are subsets of X. The cross validation is then executed as a for-loop, iterating through each X_{train} and X_{val} in F. For each run-through, an empty XGBoost ensemble, E_0, is initialized and gradually built through a for-loop with a total of C run-throughs; One for each tree. For each step in the building process, an XGBoost tree, t_c, is defined with its associated per-tree hyperparameter configuration in H, H_c. This tree is then added to the ensemble, denoted as E_c. When the building process is complete, the finalized ensemble, E_C, is trained on X_{train} and evaluated on X_{val} to obtain the prediction performance value, p, which is added to P. After H has been evaluated on all cross validation folds, the cross validation score is calculated as the average of P, and returned as the result of the Objective. The Objective is optimized through I iterations of

Bayesian Optimization, each producing a flexible ensemble structure configuration based on the search space, S, and evaluating it on the Objective. Finally, the best prediction performance, p_{best} and the associated flexible ensemble structure, H_{best}, are determined from the optimization iterations, and returned as the result of the procedure.

5.1.2 Investigating the Possibility of Overfitting

An important thing to rule out was the possibility that the results could be inaccurate due to overfitting to the cross validation folds. To investigate whether this was the case, we investigated an instance of Scenario 4 on the largest dataset, Energy Prediction, separate from the earlier, where we used a hold-out test set. We used the largest dataset to minimize the probability that the training- and test-sets would be unrepresentative of each other. Holistic optimization was implemented as with the pseudo code in Figure 5.1, with X being set to the training-set. After this procedure was completed, we built an ensemble based on the obtained flexible structure configuration and evaluated it on the test-set.

5.2 Experiment 2

Experiment 2 was conducted to obtain insight into RQ3 by comparing the prediction performances of Holistic and Incremental flexible ensemble structure optimization. To achieve this, in this experiment, we optimized flexible ensemble structures with the Incremental approach and compared their prediction performance to those obtained in Experiment 1. To ensure that the prediction performances would be reasonable to compare, the Incremental flexible ensemble structure approach was implemented with very similar factors to the Holistic approach in Experiment 1. For instance, to achieve as optimally performing configurations as possible with the Incremental structure optimization approach, we again implemented Bayesian Optimization as the optimization method. The number of ensemble trees remained 5, and the ensembles were again evaluated with 2 repetitions of 5-fold cross validation. Mean Absolute Error was used as the prediction performance metric for the regression datasets, while Error was used for the classification datasets. One flexible ensemble structure was produced with the Incremental structure optimization approach for each of the datasets outlined in Section 4.1. Each tree was optimized in isolation with 200 iterations of Bayesian Optimization. There was, however, an exception made for classification datasets, where the first tree does not by itself impact prediction performance due to XGBoost implementation details. For this reason, the first two trees were optimized together for the Car Evaluation, Statlog Satellite, and Winequality-red datasets. The investigated and compared hyperparameter for this experiment was learning_rate, equivalent to Scenario 1 of Experiment 1.

5.2.1 Flexible Ensemble Structure Procedure

The pseudo code for the flexible ensemble structure generation procedure of Experiment 2 is contained in Figure 5.2, and builds upon the pseudo code of the Incremental approach to flexible ensemble structure optimization, as outlined in Section 3.2.2. Of the procedure's input-parameters; C, I and S remain the same, respectively denoting the number of ensemble trees, the number of optimization method iterations, and the search space. X is an added input-parameter denoting the dataset, being a set of N data instances, each in the form $(\mathbf{x}, y)$.

```
procedure EXPERIMENT_2(X = {(x_1, y_1), ..., (x_N, y_N)}, C, I ∈ ℕ,
                       S = {range of (l_1, u_1), ..., range of (l_M, u_M)})
    E_0 ← empty ensemble
    P_all ← empty array
    H_all ← empty array
    for c ← 1 to C do
        procedure OBJECTIVE(H = per-tree hyperparameter configuration in
                            the form (h_1, ..., h_M), where each h is a value
                            for a unique hyperparameter)
            t ← XGBoost tree with H per-tree hyperparameter configuration
            E_ext ← t added to E_{c-1}
            P_ext ← ∅
            F ← set of cross validation folds in the form (X_train, X_val), where
            each X_train and X_val are subsets of X
            for each X_train, X_val in F do
                p_ext ← prediction performance of E_ext, trained on X_train and evaluated
                on X_val
                P_ext ← P_ext ∪ p_ext
            end for
            return average(P_ext), E_ext
        end procedure
        p_c, H_c, E_c ← the best prediction performance, per-tree hyperparameter
        configuration and extended ensemble of I iterations of Bayesian
        Optimization, each producing a per-tree hyperparameter configuration,
        H, evaluated on the Objective
        P_all ← p_c added to P_all
        H_all ← H_c added to H_all
    end for
    return P_all, H_all
end procedure
```

Figure 5.2: The pseudo code for the generation procedure of one flexible ensemble structures in Experiment 2. The procedure is based on the Incremental approach to flexible ensemble structure optimization.

The specific inputs used were as follows: X was set to the full version of the relevant dataset, C was set to 5, I was set to 200, and S contained the range for learning_rate, being 0 to 1, with values being uniformly selected decimal numbers.

As the flexible ensemble structure generation procedure of Experiment 2 is based on Incremental flexible ensemble structure optimization, the overall pseudo code is largely similar to that outlined in Figure 3.6 of Section 3.2.2. The primary differences are the use of cross validation as the evaluation method, and how returned objects are handled. The procedure begins by initializing an empty ensemble, E_0, and two empty arrays, P_{all} and H_{all}, to respectively contain the prediction performance values and per-tree hyperparameter configurations obtained after all incrementally added trees in the ensemble building process. The ensemble is then built though a for-loop of C run-thoughs. The Objective's inputparameter, H, remains unchanged, being a per-tree hyperparameter

configuration. The Objective begins by defining an XGBoost tree, t, based on the per-tree hyperparameter configuration, H, and the current ensemble, E_{c-1}, is extended with this tree. An empty set, P, is initialized to contain the prediction performance values on the different cross validation scores, the folds of which are denoted as F. For each training- and validation-fold, X_{train} and X_{val}, in F, the prediction performance, p_{ext}, based on E_{ext} trained on X_{train} and evaluated on X_{val}, is measured and added to P_{ext}. The cross validation score, the average of P_{ext}, and E_{ext} are returned as the result of the Objective. The Objective is optimized with Bayesian Optimization through I iterations, each producing a single per-tree hyperparameter configuration and evaluating it on the Objective. After all iterations, the best prediction performance, p_c, per-tree hyperparameter configuration, H_c, and extended ensemble, E_c, are defined. p_c is then added to P_{all}, and H_c is added to H_{all}. After the ensemble building process is complete, P_{all} and H_{all} are returned as the result of the procedure.

5.3 Experiment 3

Experiment 3 was conducted to obtain insight into RQ2 by investigating how the general difficulty of finding well performing configurations compares between flexible and traditional ensemble structures. To achieve this, we needed to compare a wide variety of configurations between traditional and flexible ensemble structures. Therefore, we used Holistic flexible ensemble structure optimization with Random Search to generate a set of random configurations for both traditional and flexible ensemble structures, and evaluated their prediction performance for further analysis. Specifically, we generated 1000 ensembles of 5 trees for each structure type and evaluated the prediction performance of each configuration with 2 repetitions of 5-fold cross validation on full datasets. Mean Absolute Error was used as the prediction performance metric for the regression datasets, while Accuracy was used for the classification datasets.

The experiment was executed separately for two hyperparameter scenarios on all datasets outlined in Section 4.1. These hyperparameter scenarios were:

Scenario 1: learning_rate optimized in isolation.

Scenario 2: learning_rate and max_depth optimized together.

To analyze the results, further discussed in Section 6.4, we extracted the average, best, and worst prediction performance for each structure type so these could be compared. We also generated histograms of the prediction performance values for visual comparisons. This was executed on all datasets discussed in Section 4.1.

5.3.1 Flexible Ensemble Structure Procedure

The pseudo code for the flexible ensemble structure generation procedure of Experiment 3 is contained in Figure 5.3, and builds upon the Holistic approach to flexible ensemble structure optimization, as outlined in Section 3.2.1. Of the input parameters, C, I and S remain the same, denoting the number of ensemble trees, the number of optimization method iterations, and the search space, respectively. X is an added input-parameter denoting the dataset, being a set of N data instances, each in the form ($\mathbf{x}$, y).

```
procedure EXPERIMENT_3( X = {(x_1, y_1), ..., (x_N, y_N)},
                        C, I ∈ ℕ, S = {range of (l_1, u_1), ...,
                        range of (l_M, u_M)})
    procedure OBJECTIVE( H = flexible ensemble structure configuration
                         represented as an array of C per-tree hyperparameter
                         configurations in the form (h_1, ..., h_M), where
                         each h is a value for a unique hyperparameter)
        P ← ∅
        F ← set of cross validation folds in the form (X_train, X_val), where
        each X_train and X_val are subsets of X
        for each X_train, X_val in F do
            E_0 ← empty XGBoost ensemble
            for c ← 1 to C do
                t_c ← XGBoost tree with H_c per-tree hyperparameter configuration
                E_c ← t_c added to E_{c-1}
            end for
            p ← prediction performance of E_C trained on X_train and evaluated on
            X_val
            P ← P ∪ p
        end for
        return average(P)
    end procedure
    R ← set of all cross validation scores from I iterations of
    Random Search, each producing a random flexible ensemble structure
    configuration, H, based on the search space S, and evaluating it on
    the Objective
    return R
end procedure
```

Figure 5.3: The pseudo code for the generation procedure of all flexible ensemble structures in Experiment 3. The procedure is based on the Holistic approach to flexible ensemble structure optimization.

The specific inputs used were as follows: X was set to the full version of the relevant dataset, C was set to 5, I was set to 1000, while S was set to be a combination of ranges for learning_rate, and max_depth, depending on the relevant hyperparameter scenario. The range for learning_rate was 0 to 1, with values being uniformly selected decimal numbers. And the range for max_depth was 1 to the number of features in the relevant dataset, with values being uniformly selected integers.

As Experiment 3 is based on Holistic flexible ensemble structure optimization, the overall procedure remains largely the same as the pseudo code outlined in Figure 3.4 of Section 3.2.1. The primary difference lies in the implementation of cross validation as the evaluation method, as well as the procedure's returned objects. The Objective's input-parameter, H, remains unchanged, being a flexible ensemble structure configuration in the form of an array. The Objective begins by initializing an empty set, P, to contain the prediction performance evaluated on the different cross validation folds. The training- and validation-folds are initialized in the set F, each in the form (X_{train}, X_{val}), where X_{train} and X_{val} are subsets

of X. The cross validation is then executed as a for-loop, iterating through each X_{train} and X_{val} in F. For each run-through, an empty XGBoost ensemble, E_0, is initialized and gradually built through a for-loop with a total of C run-throughs. One for each tree. For each step in the building process an XGBoost tree, t_c, is defined with its associated per-tree hyperparameter configuration in H, H_c. This tree is then added to the ensemble, denoted as E_c. When the building process is complete, the finalized ensemble, E_C, is trained on X_{train} and evaluated on X_{val} to obtain the prediction performance value, p, which is added to P. After H has been evaluated on all cross validation folds, the cross validation score is calculated as the average of P, and returned as the result of the Objective. The Objective is optimized through I iterations of Random Search, each producing a random flexible ensemble structure configuration based on the search space, S, and evaluating it on the Objective. Finally, all iterations' cross validation scores, denoted as R, are returned as the result of the procedure.

5.4 Experiment 4

Experiment 4 was conducted to obtain insight into RQ2 and RQ3 by investigating how the characteristics of flexible ensemble structure configurations might be tied to prediction performance. To achieve this we, needed to produce a wide variety of flexible structures so we could could compare their characteristics in relation to their prediction performance. For this purpose, we used Holistic flexible ensemble structure optimization with Random Search. Specifically, we generated 5000 randomly configured flexible ensemble structures of 5 trees, each trained on the datasets' dedicated training-sets and evaluated on their dedicated test-sets, as described in Section 4.1.1. Cross validation was not used for this investigation, as the "correctness" of the prediction performance values was not as important as the characteristics tied to them. The computational limitation discussed in Section 4.5, was also a considered factor. Mean Absolute Error was used as the prediction performance metric for the regression datasets, while Accuracy was used for the classification datasets. The optimized hyperparameter was learning_rate.

For the analysis of the results, we ordered the configurations based on their prediction performance, and created two groups of 10 configurations based on the best and worst prediction performance. The overall procedure was executed on all datasets discussed in Section 4.1.

5.4.1 Flexible Ensemble Structure Procedure

The pseudo code for Experiment 4 is contained in Figure 5.4, and, as with Experiment 3, builds upon Holistic flexible ensemble structure optimization, outlined in Section 3.2.1. The input parameters, C, I and S, remain the same, denoting the number of ensemble trees, the number of optimization method iterations and the search space, respectively. X_{train} and X_{test} are added pararameters for the training- and test-set, respectively, each a set of N data instances in the form ($\mathbf{x}$, y).

The specific inputs used were as follows: X_{train} and X_{test} were respectively set to the training- and test-spits of the relevant dataset, as discussed in 4.1.1. C was set to 5, I was set to 5000, while S contained the range for learning_rate, being 0 to 1, with values being uniformly selected decimal numbers.

```
procedure EXPERIMENT_4(X_train, X_test = {(x_1, y_1), ..., (x_N, y_N)},
                       C, I ∈ ℕ, S = {range of (l_1, u_1), ...,
                       range of (l_M, u_M)})
    procedure OBJECTIVE( H = flexible ensemble structure configuration
                         represented as an array of C per-tree hyperparameter
                         configurations in the form (h_1, ..., h_M), where each
                         h is a value for a unique hyperparameter)
        E_0 ← empty ensemble
        for c ← 1 to C do
            t_c ← XGBoost tree with H_c per-tree hyperparameter configuration
            E_c ← t_c added to E_{c−1}
        end for
        p ← prediction performance of E_C, trained on X_train and evaluated
        on X_test
        return p
    end procedure
    R ← array of all results of I iterations of Random Search, each
    producing a random flexible ensemble structure, H, based on the
    search space, S, evaluated on the Objective
    R_p ← array of the prediction performance values of R
    R_H ← array of the flexible ensemble structure configurations of R
    return R_p, R_H
end procedure
```

Figure 5.4: The pseudo code for the generation procedure of all flexible ensemble structures in Experiment 4. The procedure is based on the Holistic approach to flexible ensemble structure optimization.

As with Experiment 3, the procedure of Experiment 4 remains largely the same as the abstract pseudo code for Holistic flexible ensemble structure optimization. Details of evaluation and returned objects, however, differ. The Objective's input-parameter, H, is an array representing a flexible ensemble structure configuration. The Objective begins by initializing the empty ensemble, E_0, which is gradually built in the following for-loop of C run-throughs. For each step in the building process, an XGBoost tree, t_c, is defined with its associated per-tree hyperparameter configuration in H, H_c, before it is added to the ensemble. The updated ensemble is denoted as E_c. With the finalized ensemble, E_C, the prediction performance, p, is trained on X_{train} and evaluated on X_{test}. p is then returned as the result of the Objective. The Objective is optimized with Random Search through I iterations, each evaluating a random flexible ensemble structure configuration based on the search space, S. An array of the results, meaning the performance values and flexible ensemble structure configurations, of all iterations is then defined as R and further split into two arrays; R_p, and R_H. These respectively contain all iterations' prediction performance values and flexible ensemble structure configurations. Finally, R_p and R_H are returned as the result of the procedure.

5.5 Experiment 5

Experiment 5 was conducted to obtain insight into RQ3 by investigating the effects of reducing the amount of unique configurations by restricting the detail of hyperparameter values. To achieve this we implemented Holistic flexible ensemble structure optimization with Bayesian Optimization, as it was implemented in Experiment 1. We then used this implementation to compare two value selection methods native to Hyperopt. Specifically; *uniform* and *quniform*. Both methods uniformly return a random value based on lower and upper bounds. However, they differ in that quniform additionally restricts the number of decimals of returned values through a sophisticated rounding method. The formula of this method is as follows[1]:

$$\text{round}(\text{uniform}(l,\, h)\ /\ q) \times q$$

l and h are here the variables for the lower and upper bounds, and q is a number, typically between 0 and 1, that dictates the rounding of selected values.

The two value methods were compared based on prediction performance values obtained in three separate processes, each executed with both value method. Specifically, the three processes optimized the learning_rate hyperparameter for ensembles of 5 trees, with 500, 1000 and 2000 iterations of Bayesian Optimization, respectively. The reason for the three processes was to investigate whether hyperparameter values selected with quniform results in better prediction performance with less iterations, compared to uniformly selected hyperparameter values. The learning_rate search range was 0 - 1 for both uniform and quniform. For the quniform method, the rounding value, q, was set to be 0.01. In practise, this rounds selected values to two decimals, limiting the number of unique values for a given per-tree learning_rate to roughly 100. The prediction performance values were evaluated with 2 repetitions of 5 fold cross validation on the full version of all datasets discussed in Section 4.1. The results were thereby obtained by comparing the prediction performance values achieved with each value selection method, uniform and quniform, across the three optimization processes.

[1] `http://hyperopt.github.io/hyperopt/getting-started/search_spaces/`

Chapter 6

Results

This chapter contains the results of the five experiments outlined in Chapter 5. First we provide an overview of the results in Section 6.1, thereafter we present the results of the individual experiments. The results of Experiment 1 through 5 are discussed in Section 6.2 through 6.6. To save space, the results are written to abstractly discuss all explored datasets. However, the explicit results per dataset are also included in Appendix A through E for Experiment 1 through 5, respectively.

6.1 Overview

Table 6.1 shows the best prediction performances for traditional structures, Holistically optimized flexible structures, and Incrementally optimized flexible structures, each based on ensembles of 5 trees. Based on these results we can see that flexible ensemble structures obtained better prediction performance than traditional structures for 6 out of 7 datasets. This was specifically the case for the flexible structures optimized with the Holistic approach. On the other hand, the ones optimized Incrementally were considerably worse than both the Holistically optimized flexible structures and the traditional, for all datasets. This demonstrates that the Holistic approach to flexible ensemble structure optimization is more effective than the Incremental approach. To put the results into a wider perspective we also compare the results to the baselines, discussed in Section 4.1.2, and state-of-the-art prediction performance. From this we can see that the flexible ensemble structures managed to surpass the baselines for the Energy Prediction and Car Evaluation. This was, however, also the case with the traditional structures, and is thus most likely not a result of the per-tree hyperparameters. The flexible structures did not surpass any of the included state-of-the-art prediction performances. This was the expected outcome due to the small sizes of ensembles optimized in experiments. Also worth to note; The state-of-the-art values are also obtained with a variety of machine learning methods, preprocessing procedures, and methods of evaluation. It therefore difficult to accurately derive insights by comparing the thesis results to these values.

Beyond these results, we also found that it practically seemed easier to obtain good prediction performance with flexible ensemble structures, compared to traditional structures. This was specifically observed in the case of evaluating random configurations. This was a quite interesting result, considering the exponentially larger amount of unique hyperparmeter configurations with flexible structures. Additionally we found many implications of how optimization processes can potentially be made more effective by exploiting reoccurring

Thesis Main Results					Baseline		State-of-The-Art	
Dataset	Metric	Traditional	Felxible (Holistic)	Flexible (Incremental)	Method	Value	Method	Value
Concrete	MAE	3.8093	**3.4615**	4.6703	CatBoost	2.71	HENSM [1]	0.0490
Energy Prediction	MAE	33.0760	**32.7644**	44.1813	XGBoost	37.01	SVM-Radial [10]	31.3600
Housing	MAE	2.4912	**2.3143**	2.8731	CatBoost	1.75	-	-
Seoul Bike Sharing	MAE	147.64	**146.99**	171.55	LightGBM	140.60	GBM [53]	109.78
Car Evaluation	Error	**0.014181**	0.015330	0.019101	XGBoost	0.020300	Rotation Forest [15]	0.013889
Statlog Satellite	Error	0.092540	**0.091297**	0.102408	CatBoost	0.089500	-	-
Winequality-red	Error	0.325826	**0.323957**	0.356157	LightGBM	0.321900	EF-KNN (k=9) [29]	0.306000

Table 6.1: The best obtained prediction performances of traditional ensemble structures, Holistically optimized flexible structures, and Incrementally optimized flexible structures. All ensemble structures consist of 5 trees. Note that the combination of hyperparameters the prediction performances are based on, are different for each value. To put the results into a wider perspective, we have also for each dataset included the best baseline prediction performance (see Section 4.1.2), and state-of-the-art prediction performance. All included baseline prediction performances were based on the standard parameters of the given gradient boosting method. The state-of-the-art prediction performances were based on the best we could find in other research. However, we could not find any clear or comparable state-of-the-art prediction performances for the Housing, and Statlog Satellite datasets.

aspects in relation to hyperparameter configurations. Such aspects included: Optimally performing combinations of optimized hyperparameters; Common hyperparameter value ranges and value developments between trees; Common locations and a discovered leniency of the area of best performance in hyperparameter search landscapes; And the result that reducing search complexity by restricting the detail of hyperparameter values can be employed without major repercussions to prediction performance.

With that said, please refer to the following sections to read a more detailed overview of the results of each Experiment.

6.2 Experiment 1 Results

As detailed in Section 5.1, Experiment 1 was based on a comparison between flexible and traditional ensemble structures of 5 trees, each optimized with Bayesian Optimization to obtain good prediction performance. Four hyperparameter scenarios were investigated:

Scenario 1: learning_rate optimized in isolation.

Scenario 2: max_depth optimized in isolation.

Scenario 3: learning_rate, max_depth and subsample optimized together.

Scenario 4: learning_rate, max_depth, subsample and colsample_bytree optimized toghether.

The primary goal of Experiment 1 was to obtain insight into RQ1; to what extent flexible structures are beneficial to prediction performance. The results of Experiment 1 that are relevant to RQ1 are thus further referred to as the main results of this experiment. The main results were specifically obtained by observing which structure type achieved the best prediction performance for the different hyperparameter scenarios of each explored

dataset, as well as the significance of the relative percentage of improvement (see Section 5.1) obtained with the flexible ensemble structures. We considered any relative percentage of prediction performance improvement above 10% as significant, and anything less, not. As discussed in Section 5.1, we also investigated whether the results could be a due to overfitting to the cross validation folds, by training and evaluating a flexible ensemble structure on the dedicated training- and test-sets of the Energy Prediction dataset. A summary of the experiment's empirical results are contained in Table 6.2 and 6.3.

Dataset	Metrics	Scenario 1	Scenario 2	Scenario 3	Scenario 4
Concrete	T-5 MAE	4.0203	7.2300	3.8093	4.0348
	T-6 MAE	3.8706	5.7934	3.6577	3.8341
	F-5 MAE	3.7416	7.1497	**3.4615**	3.6985
	PI	186.17%	5.58%	**229.41%**	167.56%
Energy Prediction	T-5 MAE	41.8208	36.0835	33.5753	33.0760
	T-6 MAE	41.5993	34.4128	33.0812	33.0011
	F-5 MAE	41.8156	34.5437	32.8321	**32.7644**
	PI	2.34%	92.16%	150.41%	**416.02%**
Housing	T-5 MAE	2.4912	4.3459	2.4958	2.5538
	T-6 MAE	2.4593	3.4740	2.4101	2.5252
	F-5 MAE	**2.3143**	4.3119	2.3690	2.4000
	PI	**554.54%**	3.89%	147.95%	537.76%
Seoul Bike Sharing	T-5 MAE	162.05	185.91	148.42	147.64
	T-6 MAE	158.70	166.12	147.28	146.96
	F-5 MAE	159.18	184.98	147.74	**146.99**
	PI	85.67%	4.69%	59.64%	**95.58%**

Table 6.2: Empirical results from Experiment 1 for the regression datasets: MAE of the 5-tree traditional structure (T-5), the 6-tree traditional structure (T-6), the 5-tree flexible structure (F-5), and the percentage of improvement (PI) between T-5 and F-5, relative to the improvement obtained with T-6 from T-5. Prediction performance values were evaluated with 2 repeats of 5 fold cross validation.

The main results of Experiment 1 are outlined in Section 6.2.1 and 6.2.2. Generally, these demonstrate that flexible ensemble structures achieved significantly better prediction performance than traditional structures for most datasets and hyperparameter scenarios, with no indication that this was caused by overfitting. The level of significance did, however, vary between datasets and hyperparameter scenarios.

However, having generated a wide variety of well optimized flexible ensemble structures that surpassed the predictive abilities of traditional structures, we saw an opportunity for obtaining results beyond the primary goal of Experiment 1. We refer to these as the experiment's secondary results. The secondary results focused largely on obtaining insight relevant to RQ3; how flexible ensemble structures, small in size, can be effectively optimized. Useful insights for this topic typically include knowledge that can be exploited to effectivize optimization processes. We, specifically, investigated which hyperparameter combinations seemed to be best for prediction performance, by observing the frequency of which different hyperparameter scenarios achieved the best performing flexible structure; We attempted to obtain insight into reoccurring and exploitable characteristics of configurations, by comparing configurations grouped by hyperparameter scenario; And we investigated

Dataset	Metrics	Scenario 1	Scenario 2	Scenario 3	Scenario 4
Car Evaluation	T-5 Error	0.017362	0.052659	**0.014181**	0.028938
	T-6 Error	0.013021	0.048898	0.009841	0.024884
	F-5 Error	0.015339	0.054106	0.020547	0.029226
	PI	**46.60%**	-38.47%	-146.68%	-7.10%
Statlog Satellite	T-5 Error	0.102641	0.098989	0.095027	0.092540
	T-6 Error	0.099067	0.096736	0.092229	0.091142
	F-5 Error	0.099533	0.098445	0.092463	**0.091297**
	PI	86.96%	24.14%	**91.63%**	88.63%
Winequality-red	T-5 Error	0.346769	0.336783	0.327406	0.325826
	T-6 Error	0.337714	0.334283	317392	0.321144
	F-5 Error	0.343971	0.332394	0.32427	**0.323957**
	PI	30.90%	**175.55%**	31.31%	39.91%

Table 6.3: Empirical results from Experiment 1 for the classification datasets: Error of the 5-tree traditional structure (T-5), the 6-tree traditional structure (T-6), the 5-tree flexible structure (F-5), and the percentage of improvement (PI) between T-5 and F-5, relative to the improvement obtained with T-6 from T-5. Prediction performance values were evaluated with 2 repeats of 5 fold cross validation.

whether hyperparameters influenced each others behavior when optimized together, by comparing the configuration characteristics between different hyperparameter scenarios.

The secondary results of Experiment 1 are outlined in Section 6.2.3, 6.2.4 and 6.2.5. From these we generally found that Scenario 4, optimizing learning_rate, max_depth, subsample and colsample_bytree, was the best performing combination of hyperparameters, which indicates that optimizing more hyperparameters in combination is beneficial for prediction performance. We found several reoccurring aspect that have the potential of being exploited for the effectiveness of optimization. For instance, we found that learning_rate values tended to rise with later trees and were never below 0.15, max_depth values were often at the higher end of the value range, 1 to the number of dataset features, and subsample values were mostly above 0.75. We also found that the hyperparameters seemed to influence each others optimal values quite a bit, which indicates that pipeline-oriented optimization approaches are not likely to be beneficial.

6.2.1 The Prediction Performance of Flexible Ensemble Structures

The flexible structures achieved better prediction performance than the traditional structures for 6 out of the 7 datasets. For these 6 datasets, the prediction performance was additionally better in all four hyperparameter scenarios. The only dataset where the flexible structures could not surpass the traditional ones, was the classification dataset, Car Evaluation. For this dataset, only the flexible Structure in Scenario 1 achieved better prediction performance, while the rest achieved worse.

Most of the scenarios across the datasets had significant percentages of prediction performance improvement for the flexible structures. Of these, several flexible structures seemed very beneficial for prediction performance, having over 100% relative percentage of improvement. This specifically occurred in one or more scenario within each of the following 4 datasets: Concrete, Energy Prediction, Housing and Winequality-red. The

largest observed percentage of improvement was 554.54% in Scenario 1 for the Housing dataset. This was also the scenario that achieved the best prediction performance for this dataset.

There were, however, other scenarios where the percentage of improvement was not significant. This was specifically the case in Scenario 1 of the Energy Prediction dataset, and in Scenario 2 of Concrete, Housing and Seoul Bike Sharing. As mentioned earlier, there was also the instance of the Car Evaluation dataset, where prediction performances of the flexible structures were worse than the traditional for Scenario 2, 3 and 4. These add up to 7 of the 28 total unique hyperparameter optimization scenarios. Of these, 4 instances were of Scenario 2, with max_depth optimized in isolation. This could in turn indicate that flexible ensemble structures are usually not beneficial for this parameter optimized in isolation. The low significance of prediction performance improvement for the instance of Scenario 1 of the Energy Prediction dataset, was most likely a conincidence with this particular scenario. The found flexible structure configuration was here relatively similar to that of the traditional structure. In other words; the traditional structure happened to be quite fitting for this particular scenario and dataset, which in turn lowered benefit of the flexible ensemble structure. The remaining 2 scenarios are of course of Car Evaluation. This dataset was a bit of an odd case. It seemed like bayesian optimization was not as effective for optimizing the flexible structures with this dataset as with the others. For instance, the max_depth range for this dataset, was 1 to 6. Despite the much lower search complexity than with many other explored scenarios, Bayesian Optimization was not even able to obtain the same prediction performance as the traditional structure, by finding the same configuration, being 6 for all trees. Thus, these scenario instances might not fully represent the abilities of flexible ensemble structures.

6.2.2 Investigating the Possibility of Overfitting

As outlined in Section 5.1, we also investigated whether overfitting to the cross validation folds was something that could have occurred and obscured the results. This was done by optimizing an ensemble for Scenario 4 with the training-set of the Energy Prediction dataset, and evaluated with a hold out test-set. We then compared the cross validation score to the prediction performance on the test-set. The result of this was a cross validation score of 34.2314 MAE and a test score of 31.6840. From this we can see that the test-score was not significantly worse than the cross validation score, but in fact better. Thereby, there were no signs that overfitting to the cross validation folds was a factor obscuring the results.

6.2.3 Best Performing Hyperparameter Combinations

To obtain insights into which combinations of hyperparameters seemed optimal for optimization, we observed how frequently the flexible structure of the different scenarios achieved the best prediction performance on the different datasets. We found that the most frequent scenario of best prediction performance, was Scenario 4. This scenario obtained the best prediction performance for 4 of the datasets; Energy Prediction, Seoul Bike Sharing, Statlog Satellite and Winequality-red. Scenario 1 and 3 each obtained the best prediction performance for one dataset, Housing and Concrete, respectively. While the best prediction performance of Car Evaluation, was not obtained by a flexible ensemble structure. Thus, the only scenario that did not achieve the best prediction performance for any dataset, was Scenario 2.

These observations indicate most datasets benefit from more regularization, considering that Scenario 3 and 4 optimize combinations of multiple hyperparameters, while Scenario 1 and 2 optimize hyperparameters in isolation. They also indicate that learning_rate is a quite important hyperparameter to optimize, even in isolation, considering this lone hyperparameter obtained the best prediction performance for Housing. Max_depth, on the other hand, does not seem to benefit from being optimized in isolation.

6.2.4 Comparing Configurations of the Same Hyperparameter Scenario

For the purpose of obtaining insight into reoccurring and exploitable aspects of flexible structure configurations, we analyzed each scenario between the datasets and looked for similarities and differences in aspects such as the percentage of improvement with the flexible structures, the characteristics of the configurations, and value ranges of the hyperparameters. The results for each Scenario are discussed in the following sub-sections.

6.2.4.1 Scenario 1

For Scenario 1, only the flexible structure of the Energy Prediction dataset had an insignificant percentage of improvement. However, the primary reason for the low significance in this case was most likely that the optimal configuration for Scenario 1 of this dataset happened to be quite similar to that of the obtained traditional structure. Regardless, it is clear that learning_rate was typically a very beneficial hyperparameter to optimize flexibly. The fact that this scenario, consisting only of learning_rate, also obtained the best performing flexible structure for one dataset, Housing, further supports this argument.

The learning_rate values ranged between 0.19 and 1.0. This indicates that it might be reasonable to exclude values less than 0.19 in search processes, at least in the case of such small ensembles. Additionally, we observed that the learning_rate values, for 5 out of 7 datasets, somewhat appeared to be gradually rising with later trees. The relevant datasets were specifically; Concrete, Housing, Seoul Bike Sharing, Statlog Satellite, and Winequality-red. This observed behaviour is also something that can potentially be exploited in search processes, for instance by employing a bias towards this behavior.

6.2.4.2 Scenario 2

Scenario 2 overall seemed a little variable in terms of the significance of the flexible structures prediction performance improvement. For 4 out of 7 datasets, the improvement was not significantly better than the traditional structures. These datasets were specifically; Car Evaluation, Concrete, Housing, and Seoul Bike Sharing. For the remaining 3 datasets; Energy Prediction, Statlog Satellite, and Winequality-red, however, the prediction performance improvement was quite significant. This indicates that the benefit of optimizing max_depth in isolation is quite dependent on the dataset, but might have a tendency of being low. This is further exemplified by the fact that this scenario did not achieve the best prediction performance for any of the datasets.

It seemed to be a common occurrence that the max_depth values were at the higher end of the dataset-relative value range of 1 to the number of features. Especially noticeable were in the datasets, Car Evaluation, Concrete, Seoul Bike Sharing, and Winequality-red. This might be an indication that regularization is not the primary benefit of this hyperparameter,

but rather optimizing how fast the ensemble learns by increasing the trees individual predictive ability through their size.

6.2.4.3 Scenario 3

The flexible structures of Scenario 3 significantly improved prediction performance for 6 out of 7 datasets, and obtained the best performing flexible structure for the Concrete dataset. The only exception was on the Car Evaluation dataset. This result makes sense, considering the higher degree of regularization with more hyperparameters. However, it is interesting how the increased search complexity did not seem to prevent significant improvement in prediction performance.

The learning_rate values of the obtained configurations, similarly to that of Scenario 1, ranged between 0.15 and 1.0, and for 4 out of 7 datasets, the values were observed to rise with later trees. The relevant datasets were; Concrete, Energy Prediction, Housing, and Statlog Satellite. Again, these result can likely be exploited to effectivize search processes. As with Scenario 2, max_depth values were frequently observed to be at the higher end of value ranges. This was especially apparent for 4 of 7 datasets; Car Evaluation, Concrete, Energy Prediction, and Winequality-red. Subsample values were also observed to mostly be at the higher end, above 0.75, of the total value range, being 0.6 to 1.0. This indicates that smaller amounts of subsampling is generally more beneficial, which can likely be exploited to effectivize search processes.

6.2.4.4 Scenario 4

The flexible structures obtained with Scenario 4 were, as those with Scenario 3, beneficial for 6 out of 7 datasets, Car Evaluation being the only exception. Additionally, Scenario 4 obtained the flexible structure of best performance for 4 out of 7 datasets, being Energy Prediction, Seoul Bike Sharing, Statlog Satellite, and Winequality-red. This is a strong indicator that more regularization, by optimizing more hyperparameters, tend to be beneficial for prediction performance.

The learning_rate values ranged between 0.18 and 0.97, and appeared to be rising in value with later trees for 1 out of 7 datasets (Energy Prediction), most max_depth values were at the higher end of the relevant value ranges, most subsample values were higher than 0.8, while colsample_bytree typically ranged the entire value range of 0.8 to 1.0. Of course, all these are aspects that can be exploited in search processes to make them more efficient.

6.2.5 Comparing Configurations of Different Hyperparameter Scenarios

To investigate how hyperparameters influenced each others optimal values, we compared configuration characteristics between different hyperparameter scenarios. Specifically, on a per dataset basis. Generally, we found the differences between scenario configurations to be considerable. It was thus clear that the hyperparameters influenced each other when optimized together. This implies that pipeline processes that optimize one hyperparameter at a time, are most likely not going to be effective for optimizing flexible structures.

6.3 Experiment 2 Results

As detailed in Section 5.2, Experiment 2 was based on comparing flexible ensemble structures, thoroughly optimized with the Incremental approach, to ones thoroughly optimized with the Holistic approach. The goal of Experiment 2 was to obtain insight into RQ3: how

Dataset		Tree 1	Tree 2	Tree 3	Tree 4	Tree 5	Ex. 1 S1
Concrete	MAE	6.7243	5.4630	5.0550	4.7785	4.6703	3.7416
	l_r	0.9997	0.9670	0.9891	0.9970	0.8650	
Energy Prediction	MAE	45.1400	44.4539	44.2411	44.1956	44.1813	41.8156
	l_r	0.7633	0.2625	0.1708	0.0849	0.0539	
Housing	MAE	3.4046	3.0257	2.9388	2.9043	2.8731	2.3143
	l_r	0.9999	0.9220	0.6546	0.4905	0.4804	
Seoul Bike Sharing	MAE	208.34	186.14	179.42	174.29	171.55	159.18
	l_r	0.9690	0.8542	0.7885	0.8043	0.6278	

Table 6.4: From Experiment 2: The flexible structures on the regression datasets obtained with Incremental flexible structure optimization, with the MAE and learning_rate for each tree, compared to the best MAE obtained in Scenario 1 of Experiment 1. Prediction performance values were evaluated with 2 repeats of 5 fold cross validation.

flexible structures of small ensembles can be effectively optimized. This was obtained by investigating which approach to flexible ensemble structure optimization, between Holistic and Incremental optimization, was best for obtaining good prediction performance. Specifically, we thoroughly optimized a flexible ensemble structure, based on learning_rate, with the Incremental approach, and compared the achieved prediction performances to those obtained with the Holistic approach from Scenario 1 of Experiment 1. The results based on this comparison are thus the main results of Experiment 2, and are discussed in Section 6.3.1. A summary of the experiment's empirical results are contained in Table 6.4 and 6.5. In general, we found that the Incremental approach was considerably worse than both the Holistic approach and the traditional one, cementing Holistic optimization as the better of these two approaches.

Dataset		Tree 1	Tree 2	Tree 3	Tree 4	Tree 5	Ex. 1 S1
Car Evaluation	Error	-	0.043974	0.033272	0.026040	0.019101	0.015339
	l_r	0.9943	0.9949	0.9886	0.8403	0.8215	
Statlog Satellite	Error	-	0.115462	0.110800	0.105905	0.102408	0.099533
	l_r	0.7413	0.9986	0.8689	0.9856	0.9616	
Winequality-red	Error	-	0.377115	0.369919	0.360537	0.356157	0.343971
	l_r	0.5842	0.7139	0.9412	0.8153	0.9709	

Table 6.5: From Experiment 2: The flexible structures on the classification dataset, obtained with Incremental flexible structure optimization, with the Error and learning_rate for each tree, compared to the best MAE obtained in Scenario 1 of Experiment 1. Note that the Error for the fist tree is not included as Tree 1 and 2 needed to be optimized together. Prediction performance values were evaluated with 2 repeats of 5 fold cross validation.

With these results, we also saw an opportunity to observe the differences between flexible structure configurations obtained with the Holistic and Incremental approach, with

the goal of obtaining exploitable knowledge for effectivizing optimization processes. The secondary results of Experiment 2 are discussed in Section 6.3.2. Generally, we found that learning_rate values of Incrementally optimized flexible structures often seemed to decrease with later trees, and generally seemed higher than those obtained with the Holistic approach. These results thus indicate that these characteristics should be avoided in optimization processes.

6.3.1 The Best Approach to Flexible Ensemble Structure Optimization

Overall the Incremental approach to flexible ensemble structure optimization achieved clearly worse results compared to those of the Holistic approach. The obtained prediction performances were considerably worse in all of the 7 datasets. In fact, the Incremental approach achieved worse results than the traditional one for 6 of 7 datasets. The only instance were Incremental surpassed the traditional approach was with the Statlog Satellite dataset. However, even here the prediction performance increase was quite small, being only about 6.51%, relative to the difference between the 5- and 6-tree traditional ensemble structures. It was thus quite clear that Incremental flexible structure optimization, was not only completely inferior to the Holistic flexible approach, but also to the traditional one for most datasets. Thus, Holistic seems to be the go-to method for optimizing boosting ensembles flexibly.

6.3.2 Comparing the Approaches' Obtained Configurations

For the purpose of obtaining exploitable knowledge for effectivizing optimization processes of flexible ensemble structures, we compared the characteristics of obtained structures between the two explored approaches. Opposed to the Holistic approach, where learning_rate values were frequently observed to seemingly rise in values with later trees (see Section 6.2.4), the opposite was the case with the Incremental approach. The learning_rate values appeared to decrease in values with later trees for 4 out of 7 datasets. Specifically, Car Evaluation, Energy Prediction, Housing, and Seoul Bike Sharing. Additionally, the learning_rate values generally seemed higher than what observed with the configurations obtained with the Holistic approach. This was especially apparent in the datasets, Concrete, Housing, Seoul Bike Sharing, and Statlog Satellite. These are indications that the Incremental approach is too greedy, which causes convergence of prediction performance, making the optimization approach suboptimal, and arguably detrimental to prediction performance. These behaviors can therefore probably be avoided to effective optimization processes.

6.4 Experiment 3 Results

As detailed in Section 5.3, Experiment 3 was based on comparing random configurations of traditional and flexible ensemble structures. The goal of Experiment 3 was to obtain insight into RQ2: to what extent flexible structures are detrimental to optimization difficulty. The results were specifically obtained by comparing the best, worst and average prediction performance, as well as histograms, between 1000 random configurations of each traditional and flexible ensemble structures. Two hyperparameter scenarios were investigated:

Scenario 1: learning_rate optimized in isolation.

Scenario 2: learning_rate and max_depth optimized in combination

A summary of the empirical results are tabulated in Table 6.6 and 6.7, while the results are discussed in Section 6.4.1. Generally, we found that it seemed easier to obtain better prediction performance with the flexible structures, especially for the regression datasets, whereas it seemed more competitive with the traditional structures for the classification datasets.

		Scenario 1			Scenario 2		
Dataset		Avg	Best	Worst	Avg	Best	Worst
Concrete	Traditional	8.3171	4.0149	35.1962	9.2206	3.8162	35.2361
	Flexible	**4.8377**	**3.9042**	**13.8173**	**5.4038**	**3.8034**	**16.1442**
	Difference	3.4797	0.1107	21.3789	3.8168	0.0128	19.0919
Energy Prediction	Traditional	48.7335	**41.8257**	97.1359	45,133	33,8593	96,7704
	Flexible	**45.9559**	42.2148	**51.8404**	**37,1433**	**33,3772**	**61,1627**
	Difference	2.7776	-0.3891	45.2955	7.9897	0.4821	35.6077
Housing	Traditional	5.1684	2.4964	22.0247	5.434	2.4937	21.9435
	Flexible	**2.9258**	**2.4475**	**11.998**	**2.9825**	**2.4198**	**12.7407**
	Difference	2.2426	0.0489	10.0267	2.4515	0.0739	9.2028
Seoul Bike Sharing	Traditional	240.85	162.15	704.05	248.17	**150.28**	702.2
	Flexible	**174.25**	**161.36**	**361.69**	**216.19**	169.98	**456.73**
	Difference	66.6	0.79	342.36	31.98	-19.70	245.47

Table 6.6: From Experiment 3: The average, best and worst MAE scores obtained for the regression datasets, with traditional and flexible structures, in Scenario 1 and 2. The bold values mark the best performing structure type for a given type of value. The prediction performance values were evaluated with 2 repetitions of 5 fold cross validation.

		Scenario 1			Scenario 2		
Dataset		Avg	Best	Worst	Avg	Best	Worst
Car Evaluation	Traditional	95.70%	**98.26%**	92.10%	87.41%	**99.50%**	70.02%
	Flexible	**95.86%**	97.85%	**92.82%**	**91.00%**	97.16%	**70.22%**
	Difference	0.16%	-0.41%	0.72%	3.59%	-2.34%	0.20%
Statlog Satellite	Traditional	**89.03%**	89.72%	86.81%	89.07%	90.34%	74.20%
	Flexible	88.96%	**89.83%**	**87.63%**	**89.46%**	**90.40%**	**86.00%**
	Difference	-0.07%	0.11%	0.82%	0.39%	0.06%	11.80%
Winequality-red	Traditional	**63.50%**	65.04%	59.88%	62.80%	**67.32%**	54.25%
	Flexible	63.18%	**65.47%**	**60.66%**	**63.41%**	66.91%	**56.69%**
	Difference	-0.32%	0.43%	0.78%	0.61%	-0.41%	2.44%

Table 6.7: From Experiment 3: The average, best and worst Error scores obtained for the classification datasets, with traditional and flexible structures, in Scenario 1 and 2. The bold values mark the best performing structure type for a given type of value. The prediction performance values were evaluated with 2 repetitions of 5 fold cross validation.

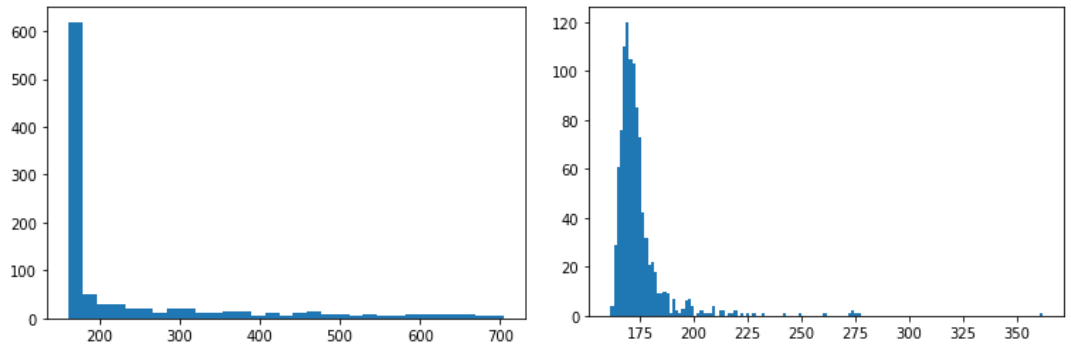
6.4.1 The Practical Search Difficulty of Flexible Ensemble Structures

The following subsections contain the results of Experiment 3 separated by the two investigated hyperparameter scenarios.

6.4.1.1 Scenario 1

For Scenario 1, the best prediction performance achieved with the flexible structures was better than those achieved with the traditional structures for 5 out of 7 datsets. These were, specifically; Concrete, Housing, Seoul Bike Sharing, Statlog Satellite and Winequality-red. The average prediction performance was also better with the flexible structures for 5 out of 7 datasets, being Concrete, Energy Prediction, Housing, Seoul Bike Sharing and Car Evaluation. While the worst obtained prediction performance was better for the flexible structures across all datasets. These are relatively strong indications that it is practically easier to obtain good prediction performance with flexible structures, compared to traditional structures, and harder to obtain bad prediction performance.

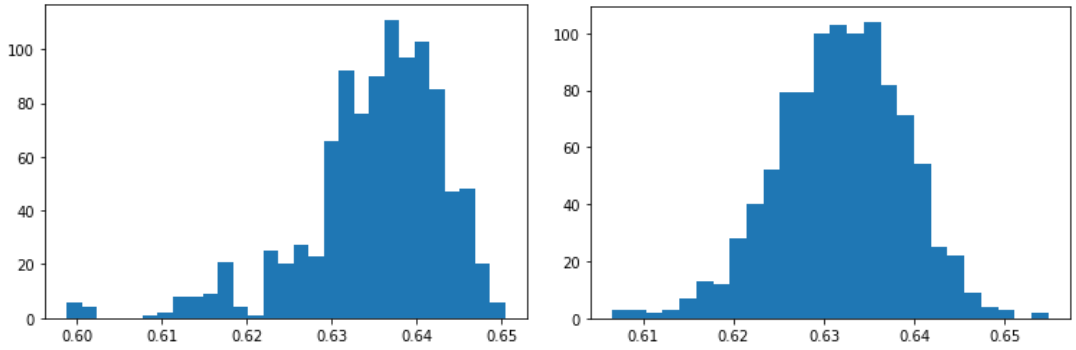
For the average prediction performances, both datasets where the flexible structures performance worse than the traditional structure were classification datasets. Additionally, the differences between the worst prediction performances of the flexible and traditional structures were smaller for the classification dataset. These could be indications that classification problems affect the search difficulty differently from regression problems, and that they are potentially harder to optimize. In terms of the histograms, the ones for the flexible structures were for all regression datasets more concentrated in a range of better performance, compared to the traditional structures, where a larger portion of values were of worse prediction performance. An example of this can be seen in Figure 6.1. This could



(a) Traditional (b) Flexible

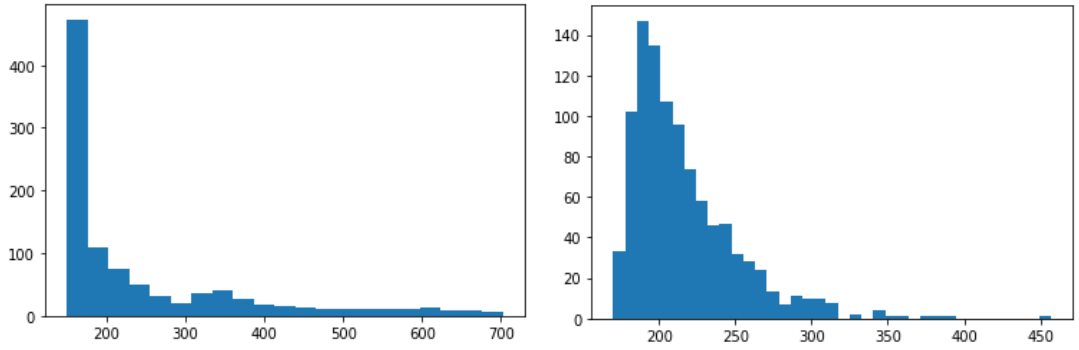
Figure 6.1: Histograms of cross validation scores from Scenario 1 of Experiment 2-1 on the Seoul Bike Sharing dataset. Figure (a) represents the cross validation scores obtained with the Traditional ensemble structure optimization approach, while Figure (b) represents the ones obtained with the Flexible approach. The horizontal axis of the histograms indicate the score values (MAE), and the vertical axis indicate the number of times these values occurred.

again indicate that obtaining bad prediction performance is harder with flexible structures, compared to traditional structures. The histograms of the classification datasets, on the other hand, were relatively similar for both structure types, the biggest difference being a slightly larger number of worse values for the traditional structures. An example can be seen in Figure 6.2.



(a) Traditional (b) Flexible

Figure 6.2: Histograms of cross validation scores from Scenario 1 of Experiment 2-1 on the Winequality-red dataset. Figure (a) represents the cross validation scores obtained with the Traditional ensemble structure optimization approach, while Figure (b) represents the ones obtained with the Flexible approach. The horizontal axis of the histograms indicate the score values (Accuracy), and the vertical axis indicate the number of times these values occurred.



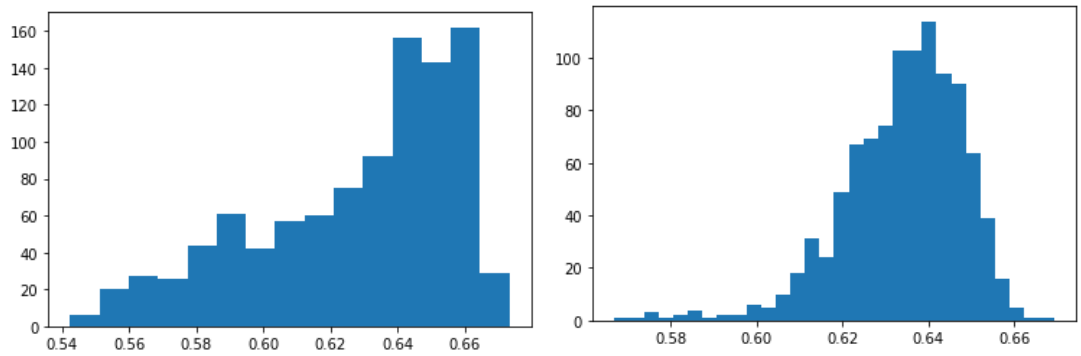
(a) Traditional (b) Flexible

Figure 6.3: Histograms of cross validation scores from Scenario 2 of Experiment 2-1 on the Seoul Bike Sharing dataset. Figure (a) represents the cross validation scores obtained with the Traditional ensemble structure optimization approach, while Figure (b) represents the ones obtained with the Flexible approach. The horizontal axis of the histograms indicate the score values (MAE), and the vertical axis indicate the number of times these values occurred.

6.4.1.2 Scenario 2

For Scenario 2, the best prediction performance of the flexible structures were better than those of the traditional structures for 4 out of 7 datasets. Namely; Concrete Energy Prediction, Housing and Statlog Satellite. The average and worst prediction performances were better for the flexible structures for all datasets. As with Scenario 1, these are relatively strong indication that it was practically easier to obtain good prediction performance

with flexible structures than with traditional ones, and harder to obtain bad prediction performance. The histograms of the prediction performances obtained with the flexible structures were for all regression datasets more concentrated in a range of better prediction performance compared to the histograms from the traditional structures (see Figure 6.3). And the histograms for the classification datasets were relatively similar between the structure types, but the traditional structures seemed to contain a slightly larger amount of worse prediction performances (see Figure 6.4). Generally, it seemed easier to obtain better



(a) Traditional (b) Flexible

Figure 6.4: Histograms of cross validation scores from Scenario 2 of Experiment 2-1 on the Winequality-red dataset. Figure (a) represents the cross validation scores obtained with the Traditional ensemble structure optimization approach, while Figure (b) represents the ones obtained with the Flexible approach. The horizontal axis of the histograms indicate the score values (Accuracy), and the vertical axis indicate the number of times these values occurred.

prediction performance with the flexible structures, especially for the regression datasets, whereas it seemed more competitive for the classification datasets.

6.5 Experiment 4 results

As detailed in Section 5.4, Experiment 4 was based on comparing flexible ensemble structure configurations of good and bad prediction performance. The goal of Experiment 4 was to obtain insight into RQ2: to what extent flexible structures are detrimental to optimization difficulty, and RQ3: how flexible structures can be effectively optimized. The results were specifically obtained by grouping the 10 best and 10 worst performing flexible ensemble structures out of 5000 random configurations based on learning_rate. We could then observe the average value and standard deviation of each tree's learning_rate.

A summary of the experiment's empirical results are tabulated Table 6.8, and the results are discussed in Section 6.5.1. Generally. we found indications that the area of best prediction performance is relatively lenient and differentiable from the area of worst prediction performance. These results not only imply that search difficulty is manageable, but are also aspects that could be exploited for optimization effectiveness.

Dataset		Tree 1	Tree 2	Tree 3	Tree 4	Tree 5
Concrete	Avg Best	0.3479	0.6568	0.8122	0.7841	0.7125
	Std Best	0.1777	0.2221	0.1467	0.1678	0.2699
	Avg Worst	0.1147	0.0959	0.1402	0.1837	0.1441
	Std Worst	0.1163	0.0815	0.0946	0.1422	0.0997
Energy Prediction	Avg Best	0.2151	0.2664	0.2595	0.2835	0.3378
	Std Best	0.0494	0.1136	0.1072	0.1163	0.0546
	Avg Worst	0.1082	0.1100	0.0974	0.1025	0.1069
	Std Worst	0.0992	0.1182	0.0833	0.1051	0.0788
Housing	Avg Best	0.2141	0.6007	0.7105	0.6470	0.7082
	Std Best	0.1081	0.2852	0.1895	0.3625	0.2289
	Avg Worst	0.1171	0.1779	0.1284	0.0949	0.1228
	Std Worst	0.0712	0.1255	0.1150	0.0737	0.0740
Seoul Bike Sharing	Avg Best	0.4614	0.6049	0.6911	0.7620	0.7723
	Std Best	0.0935	0.1486	0.1210	0.1689	0.1491
	Avg Worst	0.1561	0.1110	0.1142	0.0728	0.1723
	Std Worst	0.1047	0.1079	0.0836	0.0602	0.1206
Car Evaluation	Avg Best	0.8498	0.6755	0.5666	0.6814	0.6620
	Std Best	0.2212	0.1566	0.2530	0.1523	0.1564
	Avg Worst	0.0812	0.2305	0.1799	0.5276	0.4296
	Std Worst	0.0924	0.1286	0.1169	0.3736	0.3537
Statlog Satellite	Avg Best	0.7143	0.6543	0.8039	0.6749	0.8186
	Std Best	0.1267	0.1601	0.1232	0.2781	0.1615
	Avg Worst	0.1011	0.1553	0.3939	0.3598	0.1417
	Std Worst	0.1576	0.2581	0.4122	0.3676	0.1525
Winequality-red	Avg Best	0.6943	0.6537	0.5318	0.5526	0.5928
	Std Best	0.0728	0.3212	0.2427	0.2757	0.2888
	Avg Worst	0.1352	0.5643	0.3381	0.4550	0.4379
	Std Worst	0.0670	0.3895	0.2349	0.3271	0.3073

Table 6.8: From Experiment 4: The averages and standard deviations of each tree's learning_rate values, based the 10 best and worst configurations. Prediction performance values were obtained by training the configurations on each dataset's training-set and evaluating it on their test-set.

6.5.1 Comparison of Best and Worst Performing Configurations

The best and worst configurations were generally quite different. Specifically, the worst values typically had much lower learning_rate values for all trees. This can be demonstrated by the considerably lower tree-averages of the worst configurations. This in turn indicates that it should generally be very possible to differentiate between the areas of best and worst prediction performance. The search difficulty will therefore not be increased by conflicting areas of good and bad performance. Thereby it should be possible to avoid the area of worst performance, and encourage searches within the area of best performance.

For both the best and the worst performing flexible structure configurations obtained in Experiment 4, the characteristics of the configurations native to each group were relatively similar. This can be demonstrated by the relatively low standard deviations, which were

generally below 0.3 for the best configurations, and generally well below 0.2 for the worst. However, considering that the standard deviations were not 0, the configurations within both groups were indeed similar, but not identical. These results indicate that the area of best and worst performance are relatively lenient, and can likely be discovered or predicted, and thereby exploited in search processes. Noteworthy, however, was the observation that the standard deviations for the worst structures, were considerably lower than those of the best. This could indicate that the area of worst performance is smaller compared to the area of the best.

6.6 Experiment 5 results

As detailed in Section 5.5, Experiment 5 was based on investigating the possibility of rounding hyperparameter values to reduce search complexity. The primary goal was thus to obtain insight into RQ3: how flexible structures of small ensembles can be effectively optimized. The results were specifically obtained by comparing the prediction performance obtained with the uniform and quniform value-selection methods native to Hyperopt.

The empirical results of the experiment are tabulated in Table 6.9 and 6.10, and the results are discussed in Section 6.6.1. Generally, the results indicate that reducing search complexity by restricting hyperparameter value detail could be reasonable for achieving better prediction performance in less iterations, at least for learning_rate values, though it is not clear whether the benefit is significant.

Dataset		500 Iterations	1000 Iterations	2000 Iterations
Concrete	Uniform	3.8946	3.8281	3.8109
	Quniform	3.8416	3.8784	**3.8006**
Energy Prediction	Uniform	41.9186	41.8411	41.7953
	Quniform	41.8425	41.8388	**41.7235**
Housing	Uniform	2.3813	2.3596	2.3210
	Quniform	2.4047	2.3397	**2.3175**
Seoul Bike Sharing	Uniform	161.09	160.44	159.76
	Quniform	160.42	160.23	**159.64**

Table 6.9: From Experiment 5: The MAEs obtained with uniform and quniform, in processes of 500, 1000 and 2000 search iterations, on the regression datasets. The prediction performance values were evaluated with 2 repetitions of 5 fold cross validation.

6.6.1 Investigating the Possibility of Hyperparameter Value Rounding

In Experiment 5, the quniform value selection method was observed to obtain better performing configurations than those obtained with uniform for 5 out of 7 datasets: Concrete, Energy Prediction, Housing, Seoul Bike Sharing and Winequality-red. However, for most of the datasets, quniform still required 2000 iterations to find the best performing configurations.

For the two datasets where uniform obtained the best prediction performance, the value was only slightly worse for quniform of the Car Evaluation dataset, while seeming a bit more considerable for the Statlog Satellite dataset. Quniform additionally achieved the best prediction performance in the majority of optimization processes for most dataset.

Dataset		500 Iterations	1000 Iterations	2000 Iterations
Car Evaluation	Uniform	0.018808	0.016786	**0.015625**
	Quniform	0.018230	0.015628	0.015628
Statlog Satellite	Uniform	0.099456	0.099689	**0.099689**
	Quniform	0.100777	0.099844	0.100466
Winequality-red	Uniform	0.346778	0.344907	0.343666
	Quniform	0.343972	0.347097	**0.343045**

Table 6.10: From Experiment 5: The Errors obtained with uniform and quniform, in processes of 500, 1000 and 2000 search iterations, on the classification datasets. The prediction performance values were evaluated with 2 repetitions of 5 fold cross validation.

The most notable exception was the Statlog Satellite dataset, where quniform achieved worse prediction performance in all contained processes. Thus it seemed like quniform was generally able to achieve better prediction performance a bit more effectively than uniform, but still required a relatively large amount of search iterations. This indicates that hyperparameter value detail is not required to be that specific to achieve good prediction performance, at least not in the case of learning_rate. It is, however, not clear whether the benefit of using value selection methods like quniform is actually significant and should be utilized in search processes.

Chapter 7

Discussion

In Chapter 3, we defined the concept of per-tree hyperparameters for gradient boosting ensembles as the the term "flexible ensemble structures", and proposed two approaches to their optimization; Holistic and Incremental flexible ensemble structure optimization. The Holistic approach optimizes all per-tree hyperparameters in the same procedure, while Incremental optimizes each tree incrementally as they generated and added to the ensemble. The thesis research questions presented in Chapter 1 were investigated though five experiments. These experiments were defined in Chapter 5, all of which were based on hyperparameter optimizing XGBoost ensembles of 5 trees on 7 datasets of varying characteristics. This chapter provides a discussion around thesis by explicitly answering the research questions, discussing the relevancy of the results and acknowledging the limitations of the thesis. In Section 7.1, the research questions are restated and answered based on the experiments and their results. In Section 7.2, the results and their implications are discussed in relation to earlier research to demonstrate their relevancy. Finally, in Section 7.3, we acknowledge some limitations of the thesis. Some ideas for future work derived from these limitations are further discussed in Chapter 8.

7.1 Research Questions

This section restates and answers the research questions based on the experiments defined in Chapter 5 and their results, presented in Chapter 6.

7.1.1 RQ1

RQ1 asked the question; To what extent are per-tree hyperparameters beneficial for the prediction performance of gradient boosting ensembles? The motivation behind this question was to establish whether the unexplored approach of optimizing per-tree hyperparameters of gradient boosting ensembles, can be used to the benefit of ensemble prediction performance. The answer to RQ1 is based on the results of Experiment 1, detailed in Section 6.2. Generally, Experiment 1 was based on comparing the obtained prediction performance between flexible and traditional ensemble structures of 5 trees, each thoroughly optimized with Bayesian Optimization. To measure the significance of differences, we calculated the percentage of improvement in prediction performance achieved with the flexible structure, relative to the improvement obtained by extending and re-optimizing the traditional structure by one tree. We conducted this comparison in four separate hyperparameter scenarios:

Scenario 1: learning_rate optimized in isolation.

Scenario 2: max_depth optimized in isolation.

Scenario 3: learning_rate, max_depth and subsample optimized together.

Scenario 4: learning_rate, max_depth, subsample and colsample_bytree optimized together.

We found that the flexible structures achieved better prediction performance than the traditional structures with 6 out of the 7 datasets. With these 6 datasets, the prediction performance was additionally better in all four hyperparameter scenarios. This did, however, entail that the flexible structures obtained inferior prediction performance compared to traditional structures for one dataset, being the classification dataset, Car Evaluation. Here the flexible structures only achieved better prediction performance in Scenario 1. This result implies that flexible structures in general are capable of achieving better prediction performance than traditional structures. Additionally, the flexible structures' prediction performance improvement from the traditional structure, relative to extending the traditional structure with one tree, was significant in most scenarios across the datasets. In one or more scenarios of 4 separate datasets, the relative percentage of improvement (see Section 5.1) was above 100%. This practically means that the flexible structures in these cases were more beneficial for prediction performance than extending the traditional structure with one tree. The largest observed percentage of improvement was 554.54%. This implies that flexible ensemble structures in certain cases can be more beneficial to prediction performance than adding more trees to a traditional structure. This practically means that this approach can create ensembles that are more compact relative to prediction performance. It is, however, important to note that we do not know exactly how much more compact flexible ensembles can become, beyond that the improvement is significant.

However, there were also hyperparameter scenarios were the percentage of improvement was insignificant (less than 10%). This was specifically the case with 7 of the 28 total unique hyperparameter scenarios. Of these, 1 instance were of Scenario 1. This was, however, likely to be coincidence, as the obtained flexible structure configuration happened to be quite similar to the obtained traditional structure. 4 instances were of Scenario 2, where max_depth was optimized in isolation. This could be an indication that flexible structures are not particularly effective with this hyperparameter optimized in isolation. The 2 remaining scenario instances, and one of the Scenario 2 instances, were all of the same dataset, being Car Evaluation. This dataset seemed like a bit of an odd case, in general. As mentioned earlier, this was the only dataset were flexible structures did not outperform traditional structures. This could potentially be caused by the optimization difficulty of this particular dataset. For instance, the max_depth range for this dataset, was 1 to 6, which is much lower than with many other datasets. Despite this, Bayesian Optimization could not, in 2000 iterations, find a configuration for Scenario 2 equal to the traditional structure. Regardless of any speculation, it was clear that flexible ensemble structures did not perform well on this particular dataset.

As a part of Experiment 1, we also investigated whether the results could have been influenced by overfitting to the cross validation folds. This investigation was based on optimizing an ensemble for Scenario 4, with the training-set of the largest dataset, Energy Prediction, and evaluating it with a hold out test-set. We could then compare the cross validation score to the prediction performance on the test-set. The result was a cross validation score of 34.2314 MAE, and a test score of 31.6840. From this we can see that

the test-score was not significantly worse than the cross validation score, but in fact better. Thereby, there were no signs that overfitting to the cross validation folds was a factor obscuring the results. We did, however, only validate the possibility of overfitting on one dataset. We can therefore not be entirely certain about the other datasets, even though overfitting seems unlikely.

In summary, we can answer RQ1 by stating that optimizing per-tree hyparparmeters of gradient boosting ensembles seems significantly beneficial for prediction performance. This supports the theory that motivated the thesis; that gradient boosting ensembles would benefit from per-tree hyperparamters because each tree is given a unique prediction task. We do, however, not know if this is actually the reason for the benefits of flexible ensemble structures.

On a side note; In Section 6.1, we compared the best prediction performance values achieved with flexible ensemble structures to the best values from the baselines in Section 4.1.2 and state-of-the-art prediction performances for the different datasets [1, 10, 53, 15, 29]. The best baseline prediction performance was surpassed for two datasets. However, we determined that this was not due to the ability of flexible ensemble structures, as the same observation was made with the traditional structures of these datasets. The flexible structures did not surpass the state-of-the-art prediction performances in any of the dataset. However, this was entirely expected, considering that the ensembles were limited to 5 trees. Thus, these observations do not detract from the other results, as their focus was comparing flexible and traditional structures, not obtaining the best prediction performance in general.

7.1.2 RQ2

RQ2 asked the question; To what extent are per-tree hyperparameters detrimental to the optimization difficulty of gradient boosting ensembles? The motivation behind this question was to determine if the optimization of per-tree hyperparameters for gradient boosting ensembles is doable with a reasonable amount of computational resources. The answer to RQ2 is based on the results of Experiment 3 and 4, detailed in Section 6.4 and 6.5, respectively.

Generally, Experiment 3 was based on comparing the average, best and worst obtained prediction performance, between 1000 random configurations of each traditional and flexible ensemble structures. This comparison was made within two separate hyperparameter scenarios:

Scenario 1: learning_rate optimized in isolation.

Scenario 2: learning_rate and max_depth optimized together.

From Scenario 1 of Experiment 3, we found that the best achieved prediction performance of the flexible structures were better than those achieved with the traditional structures for 5 out of the 7 explored datasets. The same observation was made in 4 datasets of Scenario 2. The average prediction performance values were also better with the flexible structures for most explored datasets. Specifically, with 5 datasets in Scenario 1, and all datasets in Scenario 2. The worst prediction performance values were better with the flexible ensemble structures for all datasets in both Scenarios. These are relatively strong indications that it was practically easier to obtain good prediction performance with the flexible ensemble structures, relative to traditional ones. Specifically, in terms of best, average and worst value when evaluating a set of random configurations. This was somewhat of an unexpected

result, considering the curse of dimentionality [64], which makes the number of unique configurations increase exponentially with each optimized hyperparmeter. We theorize that the reason for this result is that a large number of flexible configurations have better prediction performance than those covered by traditional structures. This would in turn increase the chance of obtaining a well performing configuration when searching the flexible structure space.

In terms of the prediction performance histograms of Experiment 3, we found that the ones based on the flexible structures were more concentrated in a range of better prediction performance, compared to those of the traditional structures. The histograms of the classification datasets, on the other hand, were relatively similar for both structure types, the biggest difference being a slightly larger number of worse values for the traditional structures. This was found to be the case in both hyperparameter scenarios. These results might indicate that it is practically easier to obtain good prediction performance with flexible structures for the regression datasets, compared to the classification datasets.

Experiment 4 was based on comparing learning_rate based flexible ensemble structure configurations of good and bad prediction performance. The good and bad configurations were grouped from a pool of 5000 random flexible ensemble structures. The comparisons between the two groups were specifically based the the average learning_rate values and standard deviations of each tree. From this experiment, we found that the best and worst configurations were generally quite different in characteristics. Specifically, the worst values typically had much lower learning_rate values for all trees. This in turn indicates that it should generally be very possible to differentiate between the areas of best and worst prediction performance. The search difficulty is therefore not negatively affected by conflicting areas of good and bad performance. Thereby it should be possible to avoid the area of worst performance, and encourage searches within the area of best performance.

In summary, we can answer RQ2, by stating that utilizing per-tree hyperparameter configurations for gradient boosting algorithms is not obviously detrimental to optimization difficulty. In fact, it can with certain optimization methods comparably result in better prediction performance than traditional hyperparameter configurations in the same amount of spent resources. We found no indications of aspects that would negatively impact search performance, such as conflicting areas of good and bad prediction performance.

7.1.3 RQ3

RQ3 asked the question; How can gradient boosting ensembles with per-tree hyperparameters be effectively optimized? The motivation behind this question was to establish a standard for how per-tree hyperparameters should be optimized, as well as gain insights into aspects that could benefit such processes. The answer to RQ3 is based primarily on the results of Experiment 2, detailed in Section 6.3, with additional insights derived from the results of Experiment 1, 4 and 5, detailed in Section 6.2, 6.5 and 6.6, respectively.

Experiment 2 was based on comparing the effectiveness of the Holistic and Incremental approach to flexible ensemble structure optimization. Specifically, we compared the prediction performance of flexible ensemble structures, thoroughly optimized with the Incremental approach, to that of ones thoroughly optimized with the Holistic approach. From the results of this experiment we found that the Incremental approach achieved considerably worse prediction performance that the Holistic approach in all of the 7 explored datasets. In fact, the Incremental approach achieved worse results than the traditional approach for 6 of the 7 explored datasets. This is a quite clear indication

that the Incremental approach is not only worse than the Holistic approach, but seems detrimental to prediction performance. We theorize that this is the case because the Incremental approach is too greedy in optimization of each ensemble tree by not considering how following trees are affected.

Experiment 1 was primarily focused on providing insights into RQ1 through comparing the prediction performance of flexible and traditional ensemble structures. However, we saw an oppurtunity to also obtain insights relevant to RQ3. We, specifically investigated which hyperparameter combinations seemed to be best for prediction performance, by observing the frequency of which different hyperparameter scenarios achieved the best performing flexible structure; We investigated whether hyperparameters influenced each others behavior when optimized together, by comparing the configuration characteristics between different hyperparameter scenarios; And we attempted to obtain insight into reoccurring and exploitable characteristics of configurations, by comparing configurations grouped by hyperparameter scenario.

Based on the results of Experiment 1, we found that flexible structures of Scenario 4 were the ones most frequently achieving the best prediction performance. This scenario specifically achieved the best prediction performance for 4 of the 7 datasets. Scenario 1 and 3 got a shared second place, having the best prediction performance for one dataset each. Scenarios of the last dataset, Car Evaluation, was excluded from this particular point of investigation because the best prediction performance, was not achieved by a flexible ensemble structure. Thus, the only scenario that did not achieve the best prediction performance for any dataset, was Scenario 2. These observations indicate that most datasets benefit from more regularization, considering that Scenario 3 and 4 optimize combinations of multiple hyperparameters, while Scenario 1 and 2 optimize hyperparameters in isolation. They also indicate that learning_rate is a quite important hyperparameter to optimize, even in isolation, considering this lone hyperparameter obtained the best prediction performance for one dataset. Max_depth, on the other hand, does not seem to benefit from being optimized in isolation. This implies that optimizing flexible ensemble structures based on multiple hyperameters should probably be the standard.

Based on comparing the configuration characteristics between different hyperparameter scenarios, we found that these were considerably different from one hyperparameter scenario to another. This indicates the hyperparameters influenced each others optimal values when optimized together. This is inherently not surprising, considering that this type of behavior has been observed since the dawn of gradient boosting. For instance with learning_rate and n_estimators [23]. The implication here, is that optimizing the hyperparameters in a pipeline process is most likely not going to be effective, because these processes often assume that the hyperparameters are relatively independent.

We also found several reoccurring aspects in the configurations. For instance we found that learning_rate values often seemed to rise with with later trees. This was specifically observed in 5 datasets with Scenario 1, in 4 datasets with Scenario 3, and in 1 dataset with Scenario 4. This totals to 10 of the 21 total relevant scenarios with this hyperparameter. Learning_rate values were additionally never observed as less than 0.15. Max_depth values were observed to often be at the higher end of value ranges, while subsample values were observed to be mostly above 0.75. These findings are thereby implications that a bias towards rising learning_rate values and higher end max_depth values could be beneficial to search processes. And also that the search ranges can potentially be narrowed. The implicit value ranges from these particular results are 0.15 to 1 for learning_rate and 0.75 to 1 for

subsample. However, we should not assume that these biases and value ranged should be used without further investigation, as we know that the hyperparameters influence each other. These ranges could thus become obsolete with larger ensembles.

To gain even further insights into these types of aspects and reoccurring behaviors, we used the results of Experiment 2 to additionally analyze the characteristics of flexible structure configurations optimized with the Incremental approach. From this analysis, we observed that learning_rate values frequently appeared to decrease in values with later trees. Specifically, this was observed in in 4 of the 7 explored datasets. This is the opposite of what was observed with the Holistically optimized flexible structures from Experiment 1. From these observations, we can derive an implication that such learning_rate behaviors are indeed tied to predictive performance, and can likely be exploited in search processes.

From Experiment 4, which was based on comparing flexible ensemble structure configurations (learning_rate) of good and bad prediction performance, we found indications that well performing areas of search landscapes are relatively lenient size, and can be differentiated from the areas of bad prediction performance. This was based the observation that the configurations grouped by best prediction performance were generally quite similar in conceptualized landscape location, but were still clearly distinguishable in characteristics. Similar observations were made with with the worst values, though in a different area of the landscape and with less distinguishable characteristics. Similarity was specifically measured based on learning_rate standard deviations grouped by tree, which were generally below 0.3 for the best configurations, and below 0.2 for the worst. These observations imply that the areas of good performance can be discovered and focused on in search processes, while safely avoiding areas of bad performance.

Experiment 5 was based on investigating the possibility of rounding hyperparameter values to reduce search complexity. More specifically, we compared the prediction performance obtained with the quniform and uniform value selection methods, native to Hyperopt, in three different optimization processes of 500, 1000 and 2000 iterations. All processes optimized learning_rate in isolation. From this experiment, we found that the quniform value selection method, native to Hyperopt, obtained configurations of better prediction performance, compared to the regular uniform value selection method. This was specifically observed with 5 of the 7 explored datasets when optimizing learning_rate. However, for most of the datasets, quniform still required 2000 iterations to find the best performing configurations. This indicates that reducing search complexity by restricting hyperparameter values could be a reasonable method for achieving better prediction performance in less search iterations. However, it is unclear whether the benefits of this method are substantial. At the very least, we can say that restricting the detail of hyperparameters is not inherently detrimental to searches.

In summary, we answer RQ3 by stating that per-tree hyperparameters of gradient boosting ensembles should by standard be optimized holistically and together in the same process. We also found many indications of aspects that could help effectivize optimization processes. Such aspects include: Optimally performing combinations of optimized hyperparameters; Common hyperparameter value ranges and value developments between trees; Common locations and a discovered leniency of the area of best performance in hyperparameter search landscapes; And the result that reducing search complexity by restricting the detail of hyperparameter values can be employed without major repercussions to prediction performance.

7.2 Relevancy of the Thesis Results

To the extent of our knowledge, per-tree hyperparameters have not been previously been investigated in the context of gradient boosting. The closest research we could find was a couple of papers that investigated the benefits of promoting diversity in gradient boosting ensemble trees based on per-tree training data [51, 5]. This type of diversity was here found to be promising for prediction performance.

While not in the context of gradient boosting, per-component hyperparamters have been explored in other contexts. For instance, stacking ensembles [62] and ensembles produced with ensemble selection [11] have per-component hyperparameters as the norm, which here benefits prediction performance by promoting diversity. We can also draw a parallel to artificial neural networks, where per-layer hyperparameters greatly impact prediction performance, and thus require careful optimization [16, 2].

The results' indications that per-tree hyperparameters for gradient boosting ensembles are significantly beneficial to prediction performance is thus very much in alignment with earlier research into other contexts, where component diversity based on hyperparameters, or otherwise, positively impacts prediction performance. This demonstrates, that per-tree hyperparameters are also beneficial with gradient boosting ensembles, and is thus a useful concept that should be utilized in further research and practical applications.

We do not have a lot in insight into how per-component hyperparameters affect optimization difficulty in other contexts. It is, however, worth to mention that neural networks most likely have a relatively similar optimization difficulty, considering their often times large amount of per-layer hyperparameters. Regardless, the per-layer hyperparameters are optimized in practise, which demonstrates that the difficulty is manageable in this context [16]. We found that it can be practically easier to obtain good prediction performance with per-tree hyperparameters in gradient boosting than with traditionally handled hyperparameters. This demonstrates that the optimization difficulty of per-tree hyperparameters not only seems manageble, as with neural networks, but could potentially be easier than with traditionally handled hyperparameters. This is a strong motivation to further investigate per-tree hyperparameters, based on the potential that this approach could replace how the hyperparameters are traditionally handled.

Throughout the thesis we have investigated two approaches to the optimization of per-tree hyperparameters; Holistic and Incremental. Comparing these approaches to that of earlier research, we can see that the Holistic approach is practically quite similar to how neural networks are optimized, where all hyperparameters are handled in the same procedure [16, 2]. The Incremental approach can be compared to certain optimization approaches applied in the context of model types like stacking ensembles, where each set of per-component hyperparmameters are either optimized completely separately [18], or more relevantly, in some kind of iterative way [37]. The iterative methods have previously in these contexts had great success over optimizing all hyperparameters in the same procedure. The results for the optimization of per-tree hyperparmeters in gradient boosting ensembles did, however, not align those of these contexts. We found that the Incremental approach achieved considerably worse prediction performance than both the Holistic and the traditional optimization approaches. This demonstrates that per-tree hyperparmeter optimization in gradient boosting ensembles most likely has to be handled holistically, while incremental or iterative processes seems less beneficial. This is likely because gradient boosting ensembles consist of base-learners that directly depend on each other, which

means that you cannot optimize one base-learner without influencing the others. Stacking ensembles, on the other hand, contain base-learners that are generally quite independent [62].

By analyzing different characteristics of obtained hyperparameter configurations of flexible structures, we found several indications of aspects that can be exploited to effectivize optimization processes. Exploitable aspects of gradient boosting hyperparameters is generally quite lacking in research. There are, however, a couple of papers that have investigated standard hyperparmeters and optimal ranges of XGBoost [47, 66]. These emphasize the importance of such insights for the accessibility and effectiveness of optimization. Thus, we can say these results are important for standardizing the optimization of per-tree hyparparameters for gradient boosting ensembles, and for the design of future optimization approaches.

7.3 Limitations of the Thesis

While the results are quite positive for the usefulness of flexible ensemble structures, it is important to emphasize some of their limitations. For instance, prior to obtaining the results, we did not know how manageable search difficulty would be. Because of this, we kept the size of ensembles to 5 trees in all experiments, as a preemptive measure. The results of this thesis thus provides no direct insight into the effects of applying flexible structures with larger ensembles. It is therefore uncertain if flexible structures continue to be beneficial in such scenarios. However, we have no current reason to believe that their benefits would not persist with larger ensembles.

XGBoost was the only gradient boosting algorithm flexible ensemble structures were experimented with. Therefore, we cannot with full confidence say that flexible ensemble structures are useful in other gradient boosting variants, like LightGBM and Catboost. Though, considering the general similarites in the ensemble building procedure and types of hyperparameter between different implementations (See Section 2.2), the results seem likely to remain similar.

While the 7 datasets used in experiments were selected to promote generalizability by being varied in characteristics and prediction problem (See Section 4.1), the number and size of these datasets were relatively small. The generalizability of the results could thus still have room for improvement.

In Experiment 1, we did not use a standardized method of measuring flexible ensemble structures' improvement in prediction performance from traditional structures because this does not exist, as far as we know. We instead defined the significance based on a percentage of flexible structures' prediction performance improvement from the traditional ones, relative to the improvement achieved by extending the traditional structure by one tree. The actual significance of improvement can therefore be put into question. However, we would still argue that this measure of significance is valid, considering that adding more trees to a gradient boosting ensemble is one of the most effective and primary methods of improving their prediction performance [23].

Somewhat similarly to the previous discussed limitation, most of the results relating to exploitable aspects for the benefit of optimization effectiveness were obtained with non-standardized methods of analysis. This, coupled with the fact that most of these aspects were not computationally confirmed, means that these particular results should be taken with a grain of salt. We would, however, like to emphasise that these results were not

essential to answering any of the research questions, but primarily provided implications for standardization and further exploration of optimization strategies.

An additional thing that is worth mentioning, is the fact that cross validation overfitting was only validated to not be present for one dataset. While cross validation overfitting is generally rare, it can happen [44]. We can therefore not be entirely certain that this did not occur with the 6 remaining dataset, though we have no suspicion that it did. It would regardless not be too detrimental to the results, considering that prediction performances were primarily used to compare flexible and traditional structures. Generalizability to new data was thus not prioritized.

As a more optimistic limitation; We do not know if the best performing configurations obtained different experiments were optimized to their full potential. These were specifically obtained with 2000 iterations of Bayesian Optimization. This is a decent number of iterations with this method, but it could certainly go higher. Thereby, it is possible that the benefits of flexible ensemble structures on prediction performance are even more significant than the results demonstrate.

Finally; In this thesis, we have investigated two approaches to optimizing flexible ensemble structures. However, because of their flexible nature, the optimization of flexible ensemble structures allows for a lot of creativity in approach. Therefore, it is possible that there are other optimization approaches that could be more effective than the ones we have experimented with. Some ideas for such approaches are outlined in Chapter 8.

Chapter 8

Future Work

In this chapter, we outline some implications for future work based on the limitations outlined in Section 7.3. We also present some ideas for how the concept of flexible ensemble structures can be extended. In Section 8.1, we discuss some implications for future work based on the results. Thereafter, in Section 8.2, we present some ideas for how flexible ensemble structure can be effectively handled with larger ensemble sizes. Finally, In Section 8.3, we discuss some additional ideas for how ensemble structures can be modified.

8.1 Further Investigations Based on the Results

First and foremost, we need to ensure that the results of the thesis are generalizable. For this reason, running the experiments on more datasets, especially of larger size should hold high priority. Implementing flexible ensemble structures with other gradient boosting algorithms, like LightGBM and CatBoost, is also important to investigate whether the benefits of flexible structure extends beyond XGBoost.

Based on the results, there are also many natural points of investigation for future work. For instance, we found certain implications that the value ranges of certain hyperparameters, like learning_rate can reasonably be restricted. It should be investigated if this is actually the case, and what the potential benefits would be. We also observed reoccurring behavior in several of the explored hyperparameters: Learning_rate values often seemed to rise with later trees, max_depth values tended to be at the higher end of the explored ranges, and subsample values were often above 0.75. Attempting to effectivize search processes by implementing biases towards these behaviours should thus be investigated to document whether this is beneficial. It is also quite possible that max_depth values larger than the number of features for a given dataset could be beneficial, and larger ranged should therefore be investigated.

We also found implications that areas of best prediction performance, in the search landscapes with flexible ensemble structures, are relatively lenient and easily differentiable from the worst performing area. Considering this, methods like Adaptive Random Search [25] and Evolutionary Algorithms are likely effective for this search problem. And though they typically require a higher amount of iterations to find optimally performing configurations, compared to methods like Bayesian Optimization, they are more parallellizable. The leniency of the areas of best prediction performance was however observed by looking solely on learning_rate. Also investigating if this is the case for the other hyparparameters, is therefore important. Investigating the hyperparameters with visualization methods,

similarly to Ørebæk and Geitle [66], would also be an interesting point of investigation. The visualization method would, however, be adjusted to handle the visualization of more than two hyperparameters.

8.2 Flexible Structures with Larger Ensemble Sizes

Another point of investigation that is important, is to establish the effects of flexible structures with larger ensembles. Specifically, if this approach continues to be beneficial for prediction performance, and how the larger ensemble size affects the search difficulty. Conducting the experiments of the thesis in this problem case should thus hold high priority for future work.

While it is difficult to say exactly how the search difficulty will be affected, we expect that it will, at the very least, increase requirements for computational resources. For this reason, we preemptively outline some possible approaches that can help balance the (presumed) benefits of the flexible structures and the computational cost to obtain them.

One approach that is likely to be effective is to give higher optimization priority to earlier ensemble trees, and lower priority to later trees. This is based on the fact that individual trees added later to gradient boosting ensembles have less impact on the ensembles prediction performance, compared to earlier added trees. This can be demonstrated with the baselines in Table 4.3 and 4.4, from Section 4.1, where we can see that the differences in prediction performances from the ensembles of 1 tree to the ones of 10 trees, are considerably larger than between the ensembles of 10 trees and the ones of 100 trees. Prioritizing the optimization of earlier trees is thus natural, as the individual hyperparameter configurations for these will be much more significant for the prediction performance, and this should also considerably save computational resources. The prioritization could, for instance, be implemented with restricted values, such as with quniform of Hyperopt, or more sophisticated methods. Of course, methods based on random search are also good for such problem cases, as they naturally ignore insignificant hyperparameters [2].

A second possible approach, is to optimize the ensemble flexibly, but at a lower resolution. For example: Let us assume the size of the ensemble is 100 trees. We start by grouping the contained trees based on their position. For instance, we can create 5 groups, the first consisting of tree 1 to 20, the second, 21 to 40, etc. We can then optimize the trees within a given group traditionally, but flexibly between the groups. The search complexity of the ensemble of 100 trees from this example, would thus be equivalent to that of ensembles of 5 trees, as optimized in the thesis experiments. Variable group sizes can also be used. Of course, we do not know if such *semi-flexible ensemble structures* are viable. However, if they, they can be used for similar purposes as with the first outlined approach, where earlier trees can be optimized at a higher resolution, while later trees can be grouped.

There is a possibility that the abstract structures, referring to the general development of configurations between the trees, of small and large ensembles, are relatively similar. Under the condition that this is the case, search ranges for the configurations of large ensembles can be predicted from those of small ensembles. An approach of effectivizing the optimization of large ensembles, for a given dataset, can thereby be to first optimize a small ensemble, use it to generate a predictive graph for the abstract ensemble structure, and then base individual tree search ranges for larger ensembles around this graph.

Finally, a fourth and very hypothetical approach, with the goal of reducing time spent on generating decision tree, could be to independently generate a wide variety of trees based

on training data and hyperparameter configuration, and apply a method for assembling them into an gradient boosting ensemble, much like post hoc ensemble generation [18]. However, this approach might very well be counter productive considering that decision trees for gradient boosting are generated based on the data provided by their previous tree. The amount of unique trees that would be required to effectively do this, could therefore be unreasonably high, and the approach very possibly less effective than simply optimizing flexible structures with Bayesian Optimization.

8.3 Additional Structure Modifications

As the thesis results demonstrate that flexible structures are beneficial, at the very least for small ensembles, this opens up the possibility that modifying ensemble structures in other ways could be also be beneficial. For instance; how the trees are connected. As we know, gradient boosting ensemble trees are connected linearly, but this is not necessarily the only way they can be connected. An example of an alternative connection structure could be one of multiple paths, where each tree in the ensemble has multiple alternatives, which use could depend on the data to be predicted. This kind of *multi-path structure* would, however, also require a method of determining which path through the ensemble is optimal.

Another alternative, that could be regarded as an extended idea of the multi-path structure, could be to implement a connection structure similar to that of artificial neural networks. In such a structure each tree could practically function as singular nodes that can receive inputs combined from multiple trees, transform the input, and pass it on again to multiple following trees. Such a *neural network ensemble structure* would obviously also require implementations of strategies similar those found in artificial neural networks, such as connection weights and tree-node biases, tuned via backpropagation. Hyperparameter optimization of these structure types could be handled as they are in neural networks, as a collection of layers, or it could be handled entirely flexibly, with a unique configuration for each tree-node.

Chapter 9

Conclusion

Hyperparameter values significantly impact prediction performance with machine learning models and require unique configurations for a given task. Hyperparameter optimization is therefore an essential part of machine learning. The hyperparameters of gradient boosting decision tree ensembles are traditionally defined as one set of hyperparameter values that universally define all contained trees. Considering that each ensemble tree is added to compensate for the inaccuracies of earlier trees, each tree can be conceptualized as fitting to a unique task. Based on this, we theorized that per-tree hyperparameters for gradient boosting ensembles could be beneficial.

In this thesis, we investigated three research questions relating to per-tree hyperparameters with gradient boosting ensembles. The first research question (RQ1) asked to what degree per-tree hyperparameters are beneficial to prediction performance in order to determine their primary benefit in the context of hyperparameter optimization. The second (RQ2) attempted to determine if per-tree hyperparameters are reasonably applicable to the optimization of gradient boosting by investigating to what extent they are detrimental to optimization difficulty. The final research question (RQ3) asked how per-tree hyperparameters can be effectively optimized to obtain insights needed for standardizing their use.

We defined the concept of per-tree hyperparameters for gradient boosting ensembles as the term "flexible ensemble structures" and proposed two optimization approaches. The first approach to flexible ensemble structure optimization, "Holistic", optimizes all per-tree hyperparameters in one procedure. The second, "Incremental", optimizes the hyperparameters of each tree incrementally as they are generated in the ensemble building process.

To answer the research questions, we conducted 5 experiments that each provided their own insights. These were based on hyperparameter optimizing flexible structures of ensembles containing 5 trees. The ensembles were specifically based on the XGBoost algorithm and were for each experiment trained and evaluated on 7 datasets of varying characteristics. To investigate RQ1, we thoroughly optimized both flexible and traditional ensemble structures with Bayesian Optimization and compared their achieved prediction performances. From this, we found that flexible ensemble structures were significantly beneficial for prediction performance, as they considerably outperformed the traditional structures in 6 of the 7 datasets.

To investigate RQ2, we generated 1000 random configurations for both flexible and traditional ensemble structures and compared their average, best and worst achieved prediction performances. We also observed differences between good and bad flexible

ensemble structures based on a pool of 5000 random configurations to determine if areas of good and bad performance were conflicting. We found that no indications that the optimization of flexible ensemble structures was overwhelmingly difficult or prevented. In fact we found that the flexible ensemble structures obtained better prediction performance on average, in the best and in the worst cases for most datasets. This indicates that obtaining good prediction performance could practically be easier for this structure type in certain aspects.

RQ3 was primarily investigated by comparing Holistic and Incremental flexible ensemble structure optimization, based on achieved prediction performance. We investigated the possibility of reducing the number of unique configurations in optimization processes by restricting the value detail of hyperparameters. Specifically, we investigated rounding of hyperparameter values. We also obtained insights that could effectivize optimization processes, by analyzing the characteristics of obtained flexible ensemble structure configurations across the different experiments. We found that the Holistic approach was considerably more effective than the Incremental approach for achieving good prediction performance. In fact the Incremental approach seemed detrimental to prediction performance. We also found that reducing the number of unique configurations in search processes with hyperparameter value rounding could be employed without major repercussions. Lastly, we found many implications of exploitable aspects for optimization effectiveness. Such aspects included: Optimally performing combinations of optimized hyperparameters; common hyperparameter value ranges and value developments between trees; as well as common locations and a discovered leniency of the area of best performance in hyperparameter search landscapes.

While the experiment results were quite positive, they do have some limitations. Most predominantly; they are based exclusively on XGBoost ensembles of 5 trees. Investigating flexible ensemble structures with other gradient boosting implementations and ensemble sizes is thus recommended to ensure generalizability.

Per-component hyperparameters have previously been utilized in certain machine learning contexts with great success or due to necessity, such as with stacking ensembles and artificial neural networks. Per-tree hyperparameters with gradient boosting ensembles, however, have not been investigated prior to this thesis. Based on the experiment results, we conclude the thesis by stating that flexible ensemble structures for gradient boosting seem significantly beneficial for prediction performance while remaining manageable in optimization difficulty. As a standard approach, we suggest to optimize all per-tree hyperparameters holistically. With that said, we highly encourage further research into flexible ensemble structures to solidify their benefits with other gradient boosting implementations and ensemble sizes. Research into alternative optimization approaches is also recommended to further standardize their use.

Bibliography

[1] Panagiotis G Asteris, Athanasia D Skentou, Abidhan Bardhan, Pijush Samui, and Kypros Pilakoutas. Predicting concrete compressive strength using hybrid ensembling of surrogate machine learning models. *Cement and Concrete Research*, 145:106449, 2021.

[2] James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. *Advances in neural information processing systems*, 24, 2011.

[3] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *The Journal of Machine Learning Research*, 13(1):281–305, 2012.

[4] James Bergstra, Brent Komer, Chris Eliasmith, Dan Yamins, and David D Cox. Hyperopt: a python library for model selection and hyperparameter optimization. *Computational Science & Discovery*, 8(1):014008, 2015.

[5] Mathias Bourel, Jairo Cugliari, Yannig Goude, and Jean-Michel Poggi. Boosting Diversity in Regression Ensembles. working paper or preprint, December 2020.

[6] Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

[7] Leo Breiman. Using adaptive bagging to debias regressions. Technical report, Technical Report 547, Statistics Dept. UCB, 1999.

[8] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[9] Jason Brownlee. Why One-Hot Encode Data in Machine Learning?, July 2017. Accessed on Jan. 13. 2021. [Online]. Available: https://machinelearningmastery.com/why-one-hot-encode-data-in-machine-learning/.

[10] Luis M Candanedo, Véronique Feldheim, and Dominique Deramaix. Data driven prediction models of energy use of appliances in a low-energy house. *Energy and buildings*, 140:81–97, 2017.

[11] Rich Caruana, Alexandru Niculescu-Mizil, Geoff Crew, and Alex Ksikes. Ensemble selection from libraries of models. In *Proceedings of the twenty-first international conference on Machine learning*, page 18, 2004.

[12] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016.

[13] Marc Claesen, Jaak Simm, Dusan Popovic, Yves Moreau, and Bart De Moor. Easy hyperparameter search using optunity. *arXiv preprint arXiv:1412.1114*, 2014.

[14] Charles Darwin. *On the Origin of Species by Means of Natural Selection or the Preservation of Favored Races in the Struggle for Life*. Murray, London, 1859.

[15] Madhusmita Das and Rasmita Dash. Performance analysis of classification techniques for car data set analysis. In *2020 International Conference on Communication and Signal Processing (ICCSP)*, pages 0549–0553. IEEE, 2020.

[16] Tobias Domhan, Jost Tobias Springenberg, and Frank Hutter. Speeding up automatic hyperparameter optimization of deep neural networks by extrapolation of learning curves. In *Twenty-fourth international joint conference on artificial intelligence*, 2015.

[17] Stefan Falkner, Aaron Klein, and Frank Hutter. Bohb: Robust and efficient hyperparameter optimization at scale. In *International Conference on Machine Learning*, pages 1437–1446. PMLR, 2018.

[18] Matthias Feurer, Aaron Klein, Katharina Eggensperger, Jost Springenberg, Manuel Blum, and Frank Hutter. Efficient and robust automated machine learning. *Advances in Neural Information Processing Systems*, 2015.

[19] Yoav Freund. Boosting a weak learning algorithm by majority. *Information and computation*, 121(2):256–285, 1995.

[20] Yoav Freund, Robert Schapire, and Naoki Abe. A short introduction to boosting. *Journal-Japanese Society For Artificial Intelligence*, 14(771-780):1612, 1999.

[21] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997.

[22] Jerome Friedman, Trevor Hastie, Robert Tibshirani, et al. Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors). *The annals of statistics*, 28(2):337–407, 2000.

[23] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

[24] Jerome H Friedman. Stochastic gradient boosting. *Computational statistics & data analysis*, 38(4):367–378, 2002.

[25] Marius Geitle and Roland Olsson. A new baseline for automated hyper-parameter optimization. In *International Conference on Machine Learning, Optimization, and Data Science*, pages 521–530. Springer, 2019.

[26] Nikolaus Hansen and Andreas Ostermeier. Adapting arbitrary normal mutation distributions in evolution strategies: The covariance matrix adaptation. In *Proceedings of IEEE international conference on evolutionary computation*, pages 312–317. IEEE, 1996.

[27] Lars Hertel, Julian Collado, Peter Sadowski, and Pierre Baldi. Sherpa: hyperparameter optimization for machine learning models. *Conference on Neural Information Processing Systems*, 2018.

[28] John Henry Holland et al. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. MIT press, 1992.

[29] Juan Hu, Hong Peng, Jun Wang, and Wenping Yu. knn-p: A knn classifier optimized by p systems. *Theoretical Computer Science*, 817:55–65, 2020.

[30] Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In *International conference on learning and intelligent optimization*, pages 507–523. Springer, 2011.

[31] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. In *Advances in neural information processing systems*, pages 3146–3154, 2017.

[32] Michael Kearns and Leslie Valiant. Cryptographic limitations on learning boolean formulae and finite automata. *Journal of the ACM (JACM)*, 41(1):67–95, 1994.

[33] Michael J Kearns, Umesh Virkumar Vazirani, and Umesh Vazirani. *An introduction to computational learning theory*. MIT press, 1994.

[34] James Kennedy and Russell Eberhart. Particle swarm optimization. In *Proceedings of ICNN'95-International Conference on Neural Networks*, volume 4, pages 1942–1948. IEEE, 1995.

[35] Eisuke Kita. *Evolutionary algorithms*. InTech, 2011.

[36] Nicolas Knudde, Joachim van der Herten, Tom Dhaene, and Ivo Couckuyt. Gpflowopt: A bayesian optimization library using tensorflow. *arXiv preprint arXiv:1711.03845*, 2017.

[37] Julien-Charles Lévesque, Christian Gagné, and Robert Sabourin. Bayesian hyperparameter optimization for ensemble learning. *arXiv preprint arXiv:1605.06394*, 2016.

[38] Marius Lindauer, Katharina Eggensperger, Matthias Feurer, Stefan Falkner, Andre Biedenkapp, and Frank Hutter. SMAC v3 Project, July 2021. Accessed on July. 29. 2021. [Online]. Available: https://github.com/automl/SMAC3.

[39] Gilles Louppe, Manoj Kumar, and Holger Nahrsteadt. Bayesian optimization with skopt — scikit-optimize 0.9.dev0 documentation. Accessed on July. 29. 2021. [Online]. Available: https://scikit-optimize.github.io/dev/auto_examples/bayesian-optimization.html.

[40] Ruben Martinez-Cantin. Bayesopt: a bayesian optimization library for nonlinear optimization, experimental design and bandits. *J. Mach. Learn. Res.*, 15(1):3735–3739, 2014.

[41] Robert T McGibbon, Carlos X Hernández, Matthew P Harrigan, Steven Kearnes, Mohammad M Sultan, Stanislaw Jastrzebski, Brooke E Husic, and Vijay S Pande. Osprey: Hyperparameter optimization for machine learning. *Journal of Open Source Software*, 1(5):34, 2016.

[42] Ron Meir and Gunnar Rätsch. An introduction to boosting and leveraging. In *Advanced lectures on machine learning*, pages 118–183. Springer, 2003.

[43] Sparsh Mittal and Shraiysh Vaishay. A survey of techniques for optimizing deep learning on gpus. *Journal of Systems Architecture*, 99:101635, 2019.

[44] Andrew Y Ng et al. Preventing" overfitting" of cross-validation data. In *ICML*, volume 97, pages 245–253. Citeseer, 1997.

[45] Sampath Kumar Palaniswamy and R Venkatesan. Hyperparameters tuning of ensemble model for software effort estimation. *Journal of Ambient Intelligence and Humanized Computing*, 12(6):6579–6589, 2021.

[46] Alain Petrowski, Sana Ben-Hamida, and Zbigniew Michalewicz. *Evolutionary Algorithms: An Overview*. John Wiley & Sons, Incorporated, Somerset, UNITED STATES, 2017.

[47] Philipp Probst, Anne-Laure Boulesteix, and Bernd Bischl. Tunability: importance of hyperparameters of machine learning algorithms. *The Journal of Machine Learning Research*, 20(1):1934–1965, 2019.

[48] Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. Catboost: unbiased boosting with categorical features. In *Advances in neural information processing systems*, pages 6638–6648, 2018.

[49] Sayan Putatunda and Kiran Rama. A modified bayesian optimization based hyper-parameter tuning approach for extreme gradient boosting. In *2019 Fifteenth International Conference on Information Processing (ICINPRO)*, pages 1–6. IEEE, 2019.

[50] Zhijian Qu, Juan Xu, Zixiao Wang, Rui Chi, and Hanxin Liu. Prediction of electricity generation from a combined cycle power plant based on a stacking ensemble and its hyperparameter optimization with a grid-search method. *Energy*, 227:120309, 2021.

[51] Nestor V Queipo and Efrain Nava. A gradient boosting approach with diversity promoting measures for the ensemble of surrogates in engineering. *Structural and Multidisciplinary Optimization*, 60(4):1289–1311, 2019.

[52] G Thippa Reddy, Sweta Bhattacharya, S Siva Ramakrishnan, Chiranji Lal Chowdhary, Saqib Hakak, Rajesh Kaluri, and M Praveen Kumar Reddy. An ensemble based machine learning model for diabetic retinopathy classification. In *2020 international conference on emerging trends in information technology and engineering (ic-ETITE)*, pages 1–6. IEEE, 2020.

[53] VE Sathishkumar, Jangwoo Park, and Yongyun Cho. Using data mining techniques for bike sharing demand prediction in metropolitan city. *Computer Communications*, 153:353–366, 2020.

[54] Robert E Schapire. The strength of weak learnability. *Machine learning*, 5(2):197–227, 1990.

[55] Matthias Seeger. Gaussian processes for machine learning. *International journal of neural systems*, 14(02):69–106, 2004.

[56] Mohsen Shahhosseini, Guiping Hu, and Hieu Pham. Optimizing ensemble weights and hyperparameters of machine learning models for regression problems. *arXiv preprint arXiv:1908.05287*, 2019.

[57] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. *Advances in neural information processing systems*, 25, 2012.

[58] Rainer Storn and Kenneth Price. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, 11(4):341–359, 1997.

[59] Ryoji Tanabe and Alex Fukunaga. Success-history based parameter adaptation for differential evolution. In *2013 IEEE congress on evolutionary computation*, pages 71–78. IEEE, 2013.

[60] Florian Wenzel, Jasper Snoek, Dustin Tran, and Rodolphe Jenatton. Hyperparameter ensembles for robustness and uncertainty quantification. *arXiv preprint arXiv:2006.13570*, 2020.

[61] Martin Wistuba, Nicolas Schilling, and Lars Schmidt-Thieme. Automatic frankensteining: Creating complex ensembles autonomously. In *Proceedings of the 2017 SIAM International Conference on Data Mining*, pages 741–749. SIAM, 2017.

[62] David H Wolpert. Stacked generalization. *Neural networks*, 5(2):241–259, 1992.

[63] Yufei Xia, Chuanzhe Liu, YuYing Li, and Nana Liu. A boosted decision tree approach using bayesian hyper-parameter optimization for credit scoring. *Expert Systems with Applications*, 78:225–241, 2017.

[64] Li Yang and Abdallah Shami. On hyperparameter optimization of machine learning algorithms: Theory and practice. *Neurocomputing*, 415:295–316, 2020.

[65] Steven R Young, Derek C Rose, Thomas P Karnowski, Seung-Hwan Lim, and Robert M Patton. Optimizing deep learning hyper-parameters through an evolutionary algorithm. In *Proceedings of the Workshop on Machine Learning in High-Performance Computing Environments*, pages 1–5, 2015.

[66] Ole-Edvard Ørebæk and Marius Geitle. Exploring the hyperparameters of xgboost through 3d visualizations. In *AAAI Spring Symposium: Combining Machine Learning with Knowledge Engineering*, 2021.

Appendix A

Experiment 1 Per-Dataset Results

A.1 Concrete

This section contains the results of the comparison between traditional and flexible structures, each optimized with Bayesian Optimization, on the Concrete dataset. The results are divided into four hyperparmeter scenarios; Scenario 1: learning_rate; Scenario 2: max_depth; Scenario 3: learning_rate, max_depth and subsample; and Scenario 4: learning_rate, max_depth, subsample and colsample_bytree.

For all scenarios, the 5-tree flexible structure improved prediction performance compared to the 5-tree traditional structure. The only scenario where the improvement was of arguably low significance, was in Scenario 2, with max_depth optimized in isolation. The percentage of improvement from the 5-tree traditional structure, relative to the improvement gained with the 6-tree traditional structure, was here only 5.58%. For Scenario 1, 3 and 4, the relative percentages of improvement were 186.17%, 229.41% and 167.56%, respectively. Scenario 3 thus had the largest relative percentage of improvement, and was also the scenario that produced the best MAE of the stuctures for this dataset, being 3.4615. The best MAE of the 5-tree traditional structures was comparably 3.8093, from the same scenario.

Regarding the characteristics of the flexible structure, there were some similarities between the scenarios' optimal hyperparameter values. For instance, learning_rate values somewhat seemed to rise with later trees for Scenario 1 and 3, and max_depth values were quite frequently 8 in all relevant scenarios, which is the highest possible value of the search range. Beyond this, however, the differences between the scenarios were considerable, and it was quite apparent that the hyperparameters influenced each other's optimal values when optimized together.

A.1.1 Scenario 1

For Scenario 1 of the Concrete dataset, the traditionally optimized structures of 5 and 6 trees are tabulated in Table A.1, while the flexible structure obtained with Holistic optimization is tabulated in Table A.2. The prediction performance comparison between the traditional structures and the flexible structure is tabulated in Table A.3. From the 5-tree traditional structure, the 5-tree flexible structure improve prediction performance equivalent to 186.17% of the improvement achieved with the 6-tree traditional structure. This demonstrates that the flexible structure was very beneficial for prediction performance with the optimization scenario of learning_rate for this dataset.

In regards to the structure's characteristics, the traditional structures had quite different learning_rate values, with roughly 0.72 for the 5-tree ensemble, and 0.55 for the 6-tree ensemble. The flexible structure had values ranging between 0.34 to 0.94, and appeared to be gradually rising with later ensemble trees.

	5 Trees	6 Trees
MAE	4.0203	3.8706
Learning_rate	0.7151	0.5477

Table A.1: The MAE score and hyperparameter configuration of the traditionally structured ensembles of 5 and 6 trees, based on hyperparameter Scenario 1. The learning_rate was optimized though 1000 iterations of Bayesian Optimization, and evaluated with cross validation on the Concrete dataset.

MAE	3.7416
l_r Tree 1	0.3473
l_r Tree 2	0.6041
l_r Tree 3	0.5513
l_r Tree 4	0.8236
l_r Tree 5	0.9307

Table A.2: The MAE score and hyperparameter configuration of a flexible ensemble structure of 5 trees, based on hyperparameter Scenario 1. The learning_rate values for each tree were optimized through 2000 iterations of Bayesian Optimization, and evaluated with cross validation on the Concrete dataset.

MAE improvement with added tree	0.1497
MAE improvement with flexible structure	0.2787
Relative percentage improvement	186.17%

Table A.3: The MAE improvement, compared to the traditional structure of 5 trees, with adding another tree to the traditional structure, and with the 5 tree flexible ensemble structure, based on hyperparameter Scenario 1 on the Concrete dataset. And the percentage of improvement obtained with the flexible structure, relative to that obtained with the added tree.

A.1.2 Scenario 2

For Scenario 2 on the Concrete dataset, the traditionally optimized structures of 5 and 6 trees are tabulated in Table A.4, while the flexible structure obtained with Holistic optimization is tabulated in Table A.5. The prediction performance comparison between the traditional structures and the flexible structure is tabulated in Table A.6. From the 5-tree traditional structure, the 5-tree flexible structure improved prediction performance equivalent to 5.58% of the improvement achieved with the 6-tree traditional structure. This demonstrates that the flexible structure was only slightly beneficial for prediction performance with the optimization scenario of max_depth for this dataset.

In regards to the structure's characteristics, the traditional structures were both configured with the max_depth value, 8. For the flexible structure, the max_depth values ranged between 5 to 8.

	5 Trees	6 Trees
MAE	7.2300	5.7934
Max_depth	8	8

Table A.4: The MAE score and hyperparameter configuration of the traditionally structured ensembles of 5 and 6 trees, based on hyperparameter Scenario 2. The max_depth was optimized though 1000 iterations of Bayesian Optimization, and evaluated with cross validation on the Concrete dataset.

MAE	7.1497
m_d Tree 1	8
m_d Tree 2	7
m_d Tree 3	6
m_d Tree 4	8
m_d Tree 5	5

Table A.5: The MAE score and hyperparameter configuration of a flexible ensemble structure of 5 trees, based on hyperparameter Scenario 2. The max_depth values for each tree were optimized through 2000 iterations of Bayesian Optimization, and evaluated with cross validation on the Concrete dataset.

MAE improvement with added tree	1.4366
MAE improvement with flexible structure	0.0803
Relative percentage improvement	5.58%

Table A.6: The MAE improvement, compared to the traditional structure of 5 trees, with adding another tree to the traditional structure, and with the 5 tree flexible ensemble structure, based on hyperparameter Scenario 2 on the Concrete dataset. And the percentage of improvement obtained with the flexible structure, relative to that obtained with the added tree.

A.1.3 Scenario 3

For Scenario 3 on the Concrete dataset, the traditionally optimized structures of 5 and 6 trees are tabulated in Table A.7, while the flexible structure obtained with Holistic optimization is tabulated in Table A.8. The prediction performance comparison between the traditional structures and the flexible structure is tabulated in Table A.9. From the 5-tree traditional structure, the 5-tree flexible structure improve prediction performance equivalent to 229.41% of the improvement achieved with the 6-tree traditional structure. This demonstrates that flexible structure was very beneficial for prediction performance with the optimization scenario of learning_rate, max_depth and subsample for this dataset.

In regards to the structure's characteristics, the traditional structures were relatively similar in all hyperparameter values, with learning_rate being roughly 0.63 and 0.5,

max_depth being 8 for both structures, and subsample being roughly 1.0 and 0.99. The flexible structure had learning_rate values ranging between 0.48 to 0.94, with with the values somewhat seeming to rise with later trees. The max_depth values were 8 for all trees except Tree 2, where it was 1. The subsample values ranged between 0.71 to 1.0, with the values of the last three trees being over 0.93.

	5 Trees	6 Trees
MAE	3.8093	3.6577
learning_rate	0.6296	0.5534
max_depth	8	8
subsample	0.9996	0.9924

Table A.7: The MAE score and hyperparameter configuration of the traditionally structured ensembles of 5 and 6 trees, based on hyperparameter Scenario 3. Learning_rate, max_depth and subsample were optimized though 1000 iterations of Bayesian Optimization, and evaluated with cross validation on the Concrete dataset.

	learning_rate	max_depth	subsample
Tree 1	0.4878	8	0.8605
Tree 2	0.7776	1	0.7131
Tree 3	0.6412	8	0.9753
Tree 4	0.6899	8	0.9349
Tree 5	0.9339	8	0.9916
MAE	3.4615		

Table A.8: The MAE score and hyperparameter configuration of a flexible ensemble structure of 5 trees, based on hyperparameter Scenario 3. The learning_rate, max_depth and subsample values for each tree were optimized through 2000 iterations of Bayesian Optimization, and evaluated with cross validation on the Concrete dataset.

MAE improvement with added tree	0.1516
MAE improvement with flexible structure	0.3478
Relative percentage improvement	229.41%

Table A.9: The MAE improvement, compared to the traditional structure of 5 trees, with adding another tree to the traditional structure, and with the 5 tree flexible ensemble structure, based on hyperparameter Scenario 3 on the Concrete dataset. And the percentage of improvement obtained with the flexible structure, relative to that obtained with the added tree.

A.1.4 Scenario 4

For Scenario 4 on the Concrete dataset, the traditionally optimized structures of 5 and 6 trees are tabulated in Table A.10, while the flexible structure obtained with Holistic optimization is tabulated in Table A.11. The prediction performance comparison between the traditional structures and the flexible structure is tabulated in Table A.12. From the 5-tree traditional structure, the 5-tree flexible structure improve prediction performance

equivalent to 167.56% of the improvement achieved with the 6-tree traditional structure. This demonstrates that flexible structure was very beneficial for prediction performance with the optimization scenario of learning_rate, max_depth, subsample and colsample_bytree for this dataset.

In regards to the structure's characteristics, the traditional structures were relatively similar in all hyperparameter values. Learning_rate values were roughly 0.65 and 0.60, max_depth were 8 for both structures, subsample were roughly 0.99 and 1.0, and were roughly 0.85 and 0.82 for colsample_bytree. The flexible structure had learning_rate values were all relatively high, ranging between 0.57 to 0.92. Max_depth values were 8 for Tree 3, 4 and 5, and were 1 and 6 for Tree 1 and 2, respectively. Subsample values were relatively low for most trees, ranging from 0.62 to 0.77 for Tree 1, 2, 3 and 5, while Tree 4 was roughly 0.98. Colsample_bytree values were also relatively low, ranging from 0.80 to 0.87 for Tree 1, 2, 4 and 5, while being roughly 0.98 for Tree 3.

	5 Trees	6 Trees
MAE	4.0348	3.8341
learning_rate	0.6537	0.5988
max_depth	8	8
subsample	0.9867	0.9996
colsample_bytree	0.8477	0.8212

Table A.10: The MAE score and hyperparameter configuration of the traditionally structured ensembles of 5 and 6 trees, based on hyperparameter Scenario 4. Learning_rate, max_depth, subsample and colsample_bytree were optimized though 1000 iterations of Bayesian Optimization, and evaluated with cross validation on the Concrete dataset.

	learning_rate	max_depth	subsample	colsample_bytree
Tree 1	0.8897	1	0.6267	0.8029
Tree 2	0.5746	6	0.7141	0.8641
Tree 3	0.6331	8	0.7387	0.9838
Tree 4	0.7834	8	0.9795	0.8363
Tree 5	0.9175	8	0.7671	0.8655
MAE	3.6985			

Table A.11: The MAE score and hyperparameter configuration of a flexible ensemble structure of 5 trees, based on hyperparameter Scenario 4. The learning_rate, max_depth, subsample and colsample_bytree values for each tree were optimized through 2000 iterations of Bayesian Optimization, and evaluated with cross validation on the Concrete dataset.

MAE improvement with added tree	0.2007
MAE improvement with flexible structure	0.3363
Relative percentage improvement	167.56%

Table A.12: The MAE improvement, compared to the traditional structure of 5 trees, with adding another tree to the traditional structure, and with the 5 tree flexible ensemble structure, based on hyperparameter Scenario 4 on the Concrete dataset. And the percentage of improvement obtained with the flexible structure, relative to that obtained with the added tree.

A.2 Energy Prediction

This section contains the results of the comparison between traditional and flexible structures, each optimized with Bayesian Optimization, on the Energy Prediction dataset. The results are divided into four hyperparmeter scenarios; Scenario 1: learning_rate; Scenario 2: max_depth; Scenario 3: learning_rate, max_depth and subsample; and Scenario 4: learning_rate, max_depth, subsample and colsample_bytree.

For all scenarios, the 5-tree flexible structure improved prediction performance compared to the 5-tree traditional structure. The only scenario where the improvement was of arguable low significance, was in Scenario 1, with learning_rate optimized in isolation. The percentage of improvement from the 5-tree traditional structure, relative to the improvement gained with the 6-tree traditional structure, was here only 2.34%. For Scenario 2, 3 and 4, the relative percentages of improvement were 92.16%, 150.41% and 416.02%, respectively. Scenario 4 thus had the largest relative percentage of improvement, and was also the scenario that produced the best MAE of the stuctures for this dataset, being 32.7644. The best MAE of the 5-tree traditional structures was comparably 33.0760, from the same scenario.

Regarding the characteristics of the flexible structure, there were some similarities between the scenarios' optimal hyperparameter values. For instance, learning_rate values somewhat seemed to rise with later trees for Scenario 3 and 4, and max_depth ranged most of the possible value space, but all trees except one, or maximum two, had values at the higher end of the range. Beyond this, however, the differences between the scenarios were considerable, and it was quite apparent that the hyperparameters influenced each other's optimal values when optimized together.

A.2.1 Scenario 1

For Scenario 1 of the Energy Prediction dataset, the traditionally optimized structures of 5 and 6 trees are tabulated in Table A.13, while the flexible structure obtained with Holistic optimization is tabulated in Table A.14. The prediction performance comparison between the traditional structures and the flexible structure is tabulated in Table A.15. From the 5-tree traditional structure, the 5-tree flexible structure improve prediction performance equivalent to 2.34% of the improvement achieved with the 6-tree traditional structure. This demonstrates that the flexible structure was only slightly beneficial for prediction performance with the optimization scenario of learning_rate for this dataset.

In regards to the structure's characteristics, the traditional structures had quite similar learning_rate values, being roughly 0.27 and 0.23 for the 5-tree and 6-tree ensembles,

respectively. The flexible structure had values ranging between 0.19 and 0.32, thus all having quite low and similar values. This could explain the low percentage of improvement with the flexible structure.

	5 Trees	6 Trees
MAE	41.8208	41.5993
Learning_rate	0.2681	0.2261

Table A.13: The MAE score and hyperparameter configuration of the traditionally structured ensembles of 5 and 6 trees, based on hyperparameter Scenario 1. The learning_rate was optimized though 1000 iterations of Bayesian Optimization, and evaluated with cross validation on the Energy Prediction dataset.

MAE	41.8156
l_r Tree 1	0.2956
l_r Tree 2	0.2387
l_r Tree 3	0.3044
l_r Tree 4	0.3124
l_r Tree 5	0.1988

Table A.14: The MAE score and hyperparameter configuration of a flexible ensemble structure of 5 trees, based on hyperparameter Scenario 1. The learning_rate values for each tree were optimized through 2000 iterations of Bayesian Optimization, and evaluated with cross validation on the Energy Prediction dataset.

MAE improvement with added tree	0.2215
MAE improvement with flexible structure	0.0052
Relative percentage improvement	2.34%

Table A.15: The MAE improvement, compared to the traditional structure of 5 trees, with adding another tree to the traditional structure, and with the 5 tree flexible ensemble structure, based on hyperparameter Scenario 1 on the Energy Prediction dataset. And the percentage of improvement obtained with the flexible structure, relative to that obtained with the added tree.

A.2.2 Scenario 2

For Scenario 2 on the Energy Prediction dataset, the traditionally optimized structures of 5 and 6 trees are tabulated in Table A.16, while the flexible structure obtained with Holistic optimization is tabulated in Table A.17. The prediction performance comparison between the traditional structures and the flexible structure is tabulated in Table A.18. From the 5-tree traditional structure, the 5-tree flexible structure improved prediction performance equivalent to 92.16% of the improvement achieved with the 6-tree traditional structure. This demonstrates that the flexible structure was significantly beneficial for prediction performance with the optimization scenario of max_depth for this dataset.

In regards to the structure's characteristics, the traditional structures were configured with the max_depth values, 12 and 16, for the 5-tree and 6-tree ensembles, respectively.

For the flexible structure, the max_depth values ranged between 1 and 27, being the entire possible value range. However, only Tree 1 and 3 had values less than 21, being 1 and 10, respectively.

	5 Trees	6 Trees
MAE	36.0835	34.4128
Max_depth	12	16

Table A.16: The MAE score and hyperparameter configuration of the traditionally structured ensembles of 5 and 6 trees, based on hyperparameter Scenario 2. The max_depth was optimized though 1000 iterations of Bayesian Optimization, and evaluated with cross validation on the Energy Prediction dataset.

MAE	34.5437
m_d Tree 1	1
m_d Tree 2	21
m_d Tree 3	10
m_d Tree 4	27
m_d Tree 5	26

Table A.17: The MAE score and hyperparameter configuration of a flexible ensemble structure of 5 trees, based on hyperparameter Scenario 2. The max_depth values for each tree were optimized through 2000 iterations of Bayesian Optimization, and evaluated with cross validation on the Energy Prediction dataset.

MAE improvement with added tree	1.6707
MAE improvement with flexible structure	1.5398
Relative percentage improvement	92.16%

Table A.18: The MAE improvement, compared to the traditional structure of 5 trees, with adding another tree to the traditional structure, and with the 5 tree flexible ensemble structure, based on hyperparameter Scenario 2 on the Energy Prediction dataset. And the percentage of improvement obtained with the flexible structure, relative to that obtained with the added tree.

A.2.3 Scenario 3

For Scenario 3 on the Energy Prediction dataset, the traditionally optimized structures of 5 and 6 trees are tabulated in Table A.19, while the flexible structure obtained with Holistic optimization is tabulated in Table A.20. The prediction performance comparison between the traditional structures and the flexible structure is tabulated in Table A.21. From the 5-tree traditional structure, the 5-tree flexible structure improve prediction performance equivalent to 150.41% of the improvement achieved with the 6-tree traditional structure. This demonstrates that flexible structure was very beneficial for prediction performance with the optimization scenario of learning_rate, max_depth and subsample for this dataset.

In regards to the structure's characteristics, the traditional structures were relatively similar in most hyperparameter values, with learning_rate being roughly 0.43 for the 5-tree

ensemble and 0.41 for the 6-tree ensemble, max_depth being 26 and 20, and subsample being roughly 0.98 and 0.97. The flexible structure had learning_rate values ranging between 0.15 and 0.74, somewhat seeming to rise with later trees. The max_depth values ranged from 2 to 26. However, only Tree 3 had a value under 18. These values also had a somewhat conceptual pattern, being 18 and 20 for the two first trees, respectively, before sinking to 2 for Tree 3, and rising to 26 for Tree 4 and 5. The subsample values ranged between 0.77 and 1.0, being somewhat similar in conceptual value patterns to that of max_depth, being roughly 0.99 and 0.89 for the first two trees, sinking to roughly 0.78 for Tree 3, and rising to roughly 0.86 and 0.94 for Tree 4 and 5, respectively.

	5 Trees	6 Trees
MAE	33.5753	33.0812
learning_rate	0.4259	0.4109
max_depth	26	20
subsample	0.9839	0.9680

Table A.19: The MAE score and hyperparameter configuration of the traditionally structured ensembles of 5 and 6 trees, based on hyperparameter Scenario 3. Learning_rate, max_depth and subsample were optimized though 1000 iterations of Bayesian Optimization, and evaluated with cross validation on the Energy Prediction dataset.

	learning_rate	max_depth	subsample
Tree 1	0.1558	18	0.9904
Tree 2	0.2144	20	0.8939
Tree 3	0.6645	2	0.7769
Tree 4	0.5122	26	0.8550
Tree 5	0.7350	26	0.9380
MAE	32.8321		

Table A.20: The MAE score and hyperparameter configuration of a flexible ensemble structure of 5 trees, based on hyperparameter Scenario 3. The learning_rate, max_depth and subsample values for each tree were optimized through 2000 iterations of Bayesian Optimization, and evaluated with cross validation on the Energy Prediction dataset.

MAE improvement with added tree	0.4941
MAE improvement with flexible structure	0.7432
Relative percentage improvement	150.41%

Table A.21: The MAE improvement, compared to the traditional structure of 5 trees, with adding another tree to the traditional structure, and with the 5 tree flexible ensemble structure, based on hyperparameter Scenario 3 on the Energy Prediction dataset. And the percentage of improvement obtained with the flexible structure, relative to that obtained with the added tree.

A.2.4 Scenario 4

For Scenario 4 on the Energy Prediction dataset, the traditionally optimized structures of 5 and 6 trees are tabulated in Table A.22, while the flexible structure obtained with Holistic optimization is tabulated in Table A.23. The prediction performance comparison between the traditional structures and the flexible structure is tabulated in Table A.24. From the 5-tree traditional structure, the 5-tree flexible structure improve prediction performance equivalent to 416.02% of the improvement achieved with the 6-tree traditional structure. This demonstrates that flexible structure was very beneficial for prediction performance with the optimization scenario of learning_rate, max_depth, subsample and colsample_bytree for this dataset.

In regards to the structure's characteristics, the traditional structures were quite different in most hyperparameter values. The learning_rate values were roughly 0.43 and 0.36, the max_depth values were 19 and 22, the subsample values were roughly 1.0 and 0.97, while the colsample_bytree values were roughly 0.81 and 0.92. The flexible structure had learning_rate values ranging between 0.18 and 0.69, somewhat seeming to rise with later trees. Max_depth values ranged from 1 to 24, with Tree 2 being the only tree with a value lower than 18. Subsample values ranged between 0.68 and 1.0, with Tree 2 being the only tree with a value lower than 0.81. Colsample_by tree values ranged between 0.8 to 0.99, with Tree 5 being the only tree with a value higher than 0.87, and values somewhat appeared to be rising with later trees.

	5 Trees	6 Trees
MAE	33.0760	33.0011
learning_rate	0.4334	0.3620
max_depth	19	22
subsample	0.9994	0.9681
colsample_bytree	0.8089	0.9248

Table A.22: The MAE score and hyperparameter configuration of the traditionally structured ensembles of 5 and 6 trees, based on hyperparameter Scenario 4. Learning_rate, max_depth, subsample and colsample_bytree were optimized though 1000 iterations of Bayesian Optimization, and evaluated with cross validation on the Energy Prediction dataset.

	learning_rate	max_depth	subsample	colsample_bytree
Tree 1	0.1844	18	0.9997	0.8000
Tree 2	0.3608	1	0.6894	0.8217
Tree 3	0.3250	19	0.9912	0.8861
Tree 4	0.6609	24	0.9931	0.8699
Tree 5	0.6877	22	0.8166	0.9845
MAE	32.7644			

Table A.23: The MAE score and hyperparameter configuration of a flexible ensemble structure of 5 trees, based on hyperparameter Scenario 4. The learning_rate, max_depth, subsample and colsample_bytree values for each tree were optimized through 2000 iterations of Bayesian Optimization, and evaluated with cross validation on the Energy Prediction dataset.

MAE improvement with added tree	0.0749
MAE improvement with flexible structure	0.3116
Relative percentage improvement	416.02%

Table A.24: The MAE improvement, compared to the traditional structure of 5 trees, with adding another tree to the traditional structure, and with the 5 tree flexible ensemble structure, based on hyperparameter Scenario 4 on the Energy Prediction dataset. And the percentage of improvement obtained with the flexible structure, relative to that obtained with the added tree.

A.3 Housing

This section contains the results of the comparison between traditional and flexible structures, each optimized with Bayesian Optimization, on the Housing dataset. The results are divided into four hyperparmeter scenarios; Scenario 1: learning_rate; Scenario 2: max_depth; Scenario 3: learning_rate, max_depth and subsample; and Scenario 4: learning_rate, max_depth, subsample and colsample_bytree.

For all scenarios, the 5-tree flexible structure improved prediction performance compared to the 5-tree traditional structure. The only scenario where the improvement was of arguably low significance, was in Scenario 2, with max_depth optimized in isolation. The percentage of improvement from the 5-tree traditional structure, relative to the improvement gained with the 6-tree traditional structure, was here only 3.89%. For Scenario 1, 3 and 4, the relative percentages of improvement were 554.54%, 147.95% and 537.76%, respectively. Scenario 1 thus had the largest relative percentage of improvement, and was also the scenario that produced the best MAE of the stuctures for this dataset, being 2.3143. The best MAE of the 5-tree traditional structures was comparably 2.4912, from the same scenario.

Regarding the characteristics of the flexible structure, there were some similarities between the scenarios' optimal hyperparameter values. For instance, learning_rate values somewhat seemed to rise with later trees for Scenario 1 and 3, and max_depth, for both Scenario 3 and 4, had relatively conceptual pattern in values between the trees. Beyond this, however, the differences between the scenarios were considerable, and it was quite apparent that the hyperparameters influenced each other's optimal values when optimized together.

A.3.1 Scenario 1

For Scenario 1 of the Housing dataset, the traditionally optimized structures of 5 and 6 trees are tabulated in Table A.25, while the flexible structure obtained with Holistic optimization is tabulated in Table A.26. The prediction performance comparison between the traditional structures and the flexible structure is tabulated in Table A.27. From the 5-tree traditional structure, the 5-tree flexible structure improve prediction performance equivalent to 554.54% of the improvement achieved with the 6-tree traditional structure. This demonstrates that the flexible structure was very beneficial for prediction performance with the optimization scenario of learning_rate for this dataset.

In regards to the structure's characteristics, the traditional structures had relatively different learning_rate values, being roughly 0.67 for the 5-tree ensemble, and 0.49 for

the 6-tree ensemble. The flexible structure had values ranging between 0.19 to 0.81, and somewhat appeared to be gradually rising with later ensemble trees.

	5 Trees	6 Trees
MAE	2.4912	2.4593
Learning_rate	0.6670	0.4862

Table A.25: The MAE score and hyperparameter configuration of the traditionally structured ensembles of 5 and 6 trees, based on hyperparameter Scenario 1. The learning_rate was optimized though 1000 iterations of Bayesian Optimization, and evaluated with cross validation on the Housing dataset.

MAE	2.3143
l_r Tree 1	0.1945
l_r Tree 2	0.5608
l_r Tree 3	0.5349
l_r Tree 4	0.8080
l_r Tree 5	0.7611

Table A.26: The MAE score and hyperparameter configuration of a flexible ensemble structure of 5 trees, based on hyperparameter Scenario 1. The learning_rate values for each tree were optimized through 2000 iterations of Bayesian Optimization, and evaluated with cross validation on the Housing dataset.

MAE improvement with added tree	0.0319
MAE improvement with flexible structure	0.1769
Relative percentage improvement	554.54%

Table A.27: The MAE improvement, compared to the traditional structure of 5 trees, with adding another tree to the traditional structure, and with the 5 tree flexible ensemble structure, based on hyperparameter Scenario 1 on the Housing dataset. And the percentage of improvement obtained with the flexible structure, relative to that obtained with the added tree.

A.3.2 Scenario 2

For Scenario 2 on the Housing dataset, the traditionally optimized structures of 5 and 6 trees are tabulated in Table A.28, while the flexible structure obtained with Holistic optimization is tabulated in Table A.29. The prediction performance comparison between the traditional structures and the flexible structure is tabulated in Table A.30. From the 5-tree traditional structure, the 5-tree flexible structure improved prediction performance equivalent to 3.89% of the improvement achieved with the 6-tree traditional structure. This demonstrates that the flexible structure was only slightly beneficial for prediction performance with the optimization scenario of max_depth for this dataset.

In regards to the structure's characteristics, the traditional structures were both configured with the max_depth value, 8. For the flexible structure, the max_depth values ranged between 2 and 9.

	5 Trees	6 Trees
MAE	4.3459	3.4740
Max_depth	5	5

Table A.28: The MAE score and hyperparameter configuration of the traditionally structured ensembles of 5 and 6 trees, based on hyperparameter Scenario 2. The max_depth was optimized though 1000 iterations of Bayesian Optimization, and evaluated with cross validation on the Housing dataset.

MAE	4.3119
m_d Tree 1	9
m_d Tree 2	4
m_d Tree 3	9
m_d Tree 4	5
m_d Tree 5	2

Table A.29: The MAE score and hyperparameter configuration of a flexible ensemble structure of 5 trees, based on hyperparameter Scenario 2. The max_depth values for each tree were optimized through 2000 iterations of Bayesian Optimization, and evaluated with cross validation on the Housing dataset.

MAE improvement with added tree	0.8719
MAE improvement with flexible structure	0.034
Relative percentage improvement	3.89%

Table A.30: The MAE improvement, compared to the traditional structure of 5 trees, with adding another tree to the traditional structure, and with the 5 tree flexible ensemble structure, based on hyperparameter Scenario 2 on the Housing dataset. And the percentage of improvement obtained with the flexible structure, relative to that obtained with the added tree.

A.3.3 Scenario 3

For Scenario 3 on the Housing dataset, the traditionally optimized structures of 5 and 6 trees are tabulated in Table A.31, while the flexible structure obtained with Holistic optimization is tabulated in Table A.32. The prediction performance comparison between the traditional structures and the flexible structure is tabulated in Table A.33. From the 5-tree traditional structure, the 5-tree flexible structure improve prediction performance equivalent to 147.95% of the improvement achieved with the 6-tree traditional structure. This demonstrates that flexible structure was very beneficial for prediction performance with the optimization scenario of learning_rate, max_depth and subsample for this dataset.

In regards to the structure's characteristics, the traditional structures differed a bit in several hyperparameter values, with learning_rate being roughly 0.67 for the 5-tree ensemble and 0.49 for the 6-tree ensemble, max_depth being 6 and 7, and subsample both being roughly 0.99. The flexible structure had learning_rate values ranging between 0.39 and 0.9, with values seemingly rising with later trees. The max_depth values ranged between 4 and 11, beginning with 8 for Tree 1, then sinking to 4 before rising to 11, and

then sinking to 9 then 5 for the final tree. The subsample values ranged between 0.82 and 1.0 with Tree 2 being the only tree with a value less than 0.92.

	5 Trees	6 Trees
MAE	2.4958	2.4101
learning_rate	0.6666	0.4931
max_depth	6	7
subsample	0.9927	0.9868

Table A.31: The MAE score and hyperparameter configuration of the traditionally structured ensembles of 5 and 6 trees, based on hyperparameter Scenario 3. Learning_rate, max_depth and subsample were optimized though 1000 iterations of Bayesian Optimization, and evaluated with cross validation on the Housing dataset.

	learning_rate	max_depth	subsample
Tree 1	0.3970	8	0.9242
Tree 2	0.3961	4	0.8269
Tree 3	0.5906	11	0.9934
Tree 4	0.7686	9	0.9933
Tree 5	0.8973	5	0.9494
MAE	2.3690		

Table A.32: The MAE score and hyperparameter configuration of a flexible ensemble structure of 5 trees, based on hyperparameter Scenario 3. The learning_rate, max_depth and subsample values for each tree were optimized through 2000 iterations of Bayesian Optimization, and evaluated with cross validation on the Housing dataset.

MAE improvement with added tree	0.0857
MAE improvement with flexible structure	0.1268
Relative percentage improvement	147.95%

Table A.33: The MAE improvement, compared to the traditional structure of 5 trees, with adding another tree to the traditional structure, and with the 5 tree flexible ensemble structure, based on hyperparameter Scenario 3 on the Housing dataset. And the percentage of improvement obtained with the flexible structure, relative to that obtained with the added tree.

A.3.4 Scenario 4

For Scenario 4 on the Housing dataset, the traditionally optimized structures of 5 and 6 trees are tabulated in Table A.34, while the flexible structure obtained with Holistic optimization is tabulated in Table A.35. The prediction performance comparison between the traditional structures and the flexible structure is tabulated in Table A.36. From the 5-tree traditional structure, the 5-tree flexible structure improve prediction performance equivalent to 537.76% of the improvement achieved with the 6-tree traditional structure. This demonstrates that flexible structure was very beneficial for prediction performance with

the optimization scenario of learning_rate, max_depth, subsample and colsample_bytree for this dataset.

In regards to the structure's characteristics, the traditional structures were relatively similar in all hyperparameter values except max_depth. The learning_rate values were roughly 0.56 and 0.54, the max_depth values were 5 and 10, subsample values were roughly 0.98 and 0.96, and colsample_bytree values were roughly 9.5 and 0.93. The flexible structure had learning_rate values ranging between 0.33 and 0.91. Max_depth values ranged between 5 and 12, and had a somewhat symetrical pattern between the trees; being 7 for the first tree, rising to 12, sinking to 10 before rising to 12 again, and finally sinking to 5 for the last tree. Subsample values ranged between 0.89 and 1.0. Colsample_bytree values ranged between 0.81 and 0.95.

	5 Trees	6 Trees
MAE	2.5538	2.5252
learning_rate	0.5612	0.5391
max_depth	5	10
subsample	0.9827	0.9633
colsample_bytree	0.9468	0.9265

Table A.34: The MAE score and hyperparameter configuration of the traditionally structured ensembles of 5 and 6 trees, based on hyperparameter Scenario 4. Learning_rate, max_depth, subsample and colsample_bytree were optimized though 1000 iterations of Bayesian Optimization, and evaluated with cross validation on the Housing dataset.

	learning_rate	max_depth	subsample	colsample_bytree
Tree 1	0.4715	7	0.9920	0.8197
Tree 2	0.6227	12	0.9457	0.9491
Tree 3	0.3326	10	0.9158	0.8954
Tree 4	0.3781	12	0.9005	0.9210
Tree 5	0.9041	5	0.8938	0.8471
MAE	2.4000			

Table A.35: The MAE score and hyperparameter configuration of a flexible ensemble structure of 5 trees, based on hyperparameter Scenario 4. The learning_rate, max_depth, subsample and colsample_bytree values for each tree were optimized through 2000 iterations of Bayesian Optimization, and evaluated with cross validation on the Housing dataset.

MAE improvement with added tree	0.0286
MAE improvement with flexible structure	0.1538
Relative percentage improvement	537.76%

Table A.36: The MAE improvement, compared to the traditional structure of 5 trees, with adding another tree to the traditional structure, and with the 5 tree flexible ensemble structure, based on hyperparameter Scenario 4 on the Housing dataset. And the percentage of improvement obtained with the flexible structure, relative to that obtained with the added tree.

A.4 Seoul Bike Sharing

This section contains the results of the comparison between traditional and flexible structures, each optimized with Bayesian Optimization, on the Seoul Bike Sharing dataset. The results are divided into four hyperparmeter scenarios; Scenario 1: learning_rate; Scenario 2: max_depth; Scenario 3: learning_rate, max_depth and subsample; and Scenario 4: learning_rate, max_depth, subsample and colsample_bytree.

For all scenarios, the 5-tree flexible structure improved prediction performance compared to the 5-tree traditional structure. The only scenario where the improvement was of arguably low significance, was in Scenario 2, with max_depth optimized in isolation. The percentage of improvement from the 5-tree traditional structure, relative to the improvement gained with the 6-tree traditional structure, was here only 4.69%. For Scenario 1, 3 and 4, the relative percentages of improvement were 85.67%, 59.64% and 95.58%, respectively. Scenario 4 thus had the largest relative percentage of improvement, and was also the scenario that produced the best MAE of a flexible structure for this dataset, being 146.99. The best MAE of the 5-tree traditional structures was comparably 147.64, from the same scenario.

Regarding the characteristics of the flexible structures, there were very considerable differences between the scenarios' optimal hyperparameter values. However, there were some minor re-occurrences, like how for learning_rate, Tree 1, 2, 3 and 4 seemed to often be at the lower half of the 0 to 1 value range, while Tree 5 always was at the higher end, and how subsample values always were above 0.8. Regardless, it was obvious that the hyperparameters influenced each other's optimal values when optimized together.

A.4.1 Scenario 1

For Scenario 1 of the Seoul Bike Sharing dataset, the traditionally optimized structures of 5 and 6 trees are tabulated in Table A.37, while the flexible structure obtained with Holistic optimization is tabulated in Table A.38. The prediction performance comparison between the traditional structures and the flexible structure is tabulated in Table A.39. From the 5-tree traditional structure, the 5-tree flexible structure improved prediction performance equivalent to 85.67% of the improvement achieved with the 6-tree traditional structure. This demonstrates that the flexible structure was significantly beneficial for prediction performance with the optimization scenario of learning_rate for this dataset.

In regards to the structure's characteristics, the traditional structures were quite similarilly configured, with learning_rate values of roughly 0.56 and 0.54. The flexible structure had values ranging between 0.49 to 0.82, and somewhat appeared to gradually rise with later ensemble trees. However, Tree 2 to 3 were notably of very similar values, roughly around 0.6.

	5 Trees	6 Trees
MAE	162.05	158.70
Learning_rate	0.5638	0.5367

Table A.37: The MAE score and hyperparameter configuration of the traditionally structured ensembles of 5 and 6 trees, based on hyperparameter Scenario 1. The learning_rate was optimized though 1000 iterations of Bayesian Optimization, and evaluated with cross validation on the Seoul Bike Sharing dataset.

MAE	159.18
l_r Tree 1	0.4953
l_r Tree 2	0.6034
l_r Tree 3	0.6115
l_r Tree 4	0.6115
l_r Tree 5	0.8170

Table A.38: The MAE score and hyperparameter configuration of a flexible ensemble structure of 5 trees, based on hyperparameter Scenario 1. The learning_rate values for each tree were optimized through 2000 iterations of Bayesian Optimization, and evaluated with cross validation on the Seoul Bike Sharing dataset.

MAE improvement with added tree	3.35
MAE improvement with flexible structure	2.87
Relative percentage improvement	85.67%

Table A.39: The MAE improvement, compared to the traditional structure of 5 trees, with adding another tree to the traditional structure, and with the 5 tree flexible ensemble structure, based on hyperparameter Scenario 1 on the Seoul Bike Sharing dataset. And the percentage of improvement obtained with the flexible structure, relative to that obtained with the added tree.

A.4.2 Scenario 2

For Scenario 2 on the Seoul Bike Sharing dataset, the traditionally optimized structures of 5 and 6 trees are tabulated in Table A.40, while the flexible structure obtained with Holistic optimization is tabulated in Table A.41. The prediction performance comparison between the traditional structures and the flexible structure is tabulated in Table A.42. From the 5-tree traditional structure, the 5-tree flexible structure improved prediction performance equivalent to 4.69% of the improvement achieved with the 6-tree traditional structure. This demonstrates that the flexible structure was only slightly beneficial for prediction performance with the optimization scenario of max_depth for this dataset.

In regards to the structure's characteristics, the traditional structures were both configured with the max_depth value, 10. The flexible structure was similarly configured close to this value, with max_depth values ranging between 9 to 11.

	5 Trees	6 Trees
MAE	185.91	166.12
Max_depth	10	10

Table A.40: The MAE score and hyperparameter configuration of the traditionally structured ensembles of 5 and 6 trees, based on hyperparameter Scenario 2. The max_depth was optimized though 1000 iterations of Bayesian Optimization, and evaluated with cross validation on the Seoul Bike Sharing dataset.

MAE	184.98
m_d Tree 1	10
m_d Tree 2	11
m_d Tree 3	10
m_d Tree 4	9
m_d Tree 5	10

Table A.41: The MAE score and hyperparameter configuration of a flexible ensemble structure of 5 trees, based on hyperparameter Scenario 2. The max_depth values for each tree were optimized through 2000 iterations of Bayesian Optimization, and evaluated with cross validation on the Seoul Bike Sharing dataset.

MAE improvement with added tree	19.79
MAE improvement with flexible structure	0.93
Relative percentage improvement	4.69%

Table A.42: The MAE improvement, compared to the traditional structure of 5 trees, with adding another tree to the traditional structure, and with the 5 tree flexible ensemble structure, based on hyperparameter Scenario 2 on the Seoul Bike Sharing dataset. And the percentage of improvement obtained with the flexible structure, relative to that obtained with the added tree.

A.4.3 Scenario 3

For Scenario 3 on the Seoul Bike Sharing dataset, the traditionally optimized structures of 5 and 6 trees are tabulated in Table A.43, while the flexible structure obtained with Holistic optimization is tabulated in Table A.44. The prediction performance comparison between the traditional structures and the flexible structure is tabulated in Table A.45. From the 5-tree traditional structure, the 5-tree flexible structure improve prediction performance equivalent to 59.64% of the improvement achieved with the 6-tree traditional structure. This demonstrates that flexible structure was significantly beneficial for prediction performance with the optimization scenario of learning_rate, max_depth and subsample for this dataset.

In regards to the structure's characteristics, the traditional structures were quite similar in values across all hyperparameters, with learning_rates being roughly 0.49 and 0.46, max_depth being 11 for both structures, and subsample values being roughly 0.94 and 0.98. The flexible structure had learning_rate values ranging between 0.31 to 0.99 with Tree 1, 3 and 4 being relatively close to 0.34. Max_depth values ranged from 9 to 14, starting and ending with 9, and with values at the higher end of the range for tree 2, 3 and 4. Subsample values ranged from 0.81 to 0.97.

	5 Trees	6 Trees
MAE	148.42	147.28
learning_rate	0.4948	0.4639
max_depth	11	11
subsample	0.9373	0.9769

Table A.43: The MAE score and hyperparameter configuration of the traditionally structured ensembles of 5 and 6 trees, based on hyperparameter Scenario 3. Learning_rate, max_depth and subsample were optimized though 1000 iterations of Bayesian Optimization, and evaluated with cross validation on the Seoul Bike Sharing dataset.

	learning_rate	max_depth	subsample
Tree 1	0.3149	9	0.8415
Tree 2	0.4447	11	0.9685
Tree 3	0.3442	14	0.9242
Tree 4	0.3671	14	0.8126
Tree 5	0.9827	9	0.9324
MAE	147.74		

Table A.44: The MAE score and hyperparameter configuration of a flexible ensemble structure of 5 trees, based on hyperparameter Scenario 3. The learning_rate, max_depth and subsample values for each tree were optimized through 2000 iterations of Bayesian Optimization, and evaluated with cross validation on the Seoul Bike Sharing dataset.

MAE improvement with added tree	1.14
MAE improvement with flexible structure	0.68
Relative percentage improvement	59.64%

Table A.45: The MAE improvement, compared to the traditional structure of 5 trees, with adding another tree to the traditional structure, and with the 5 tree flexible ensemble structure, based on hyperparameter Scenario 3 on the Seoul Bike Sharing dataset. And the percentage of improvement obtained with the flexible structure, relative to that obtained with the added tree.

A.4.4 Scenario 4

For Scenario 4 on the Seoul Bike Sharing dataset, the traditionally optimized structures of 5 and 6 trees are tabulated in Table A.46, while the flexible structure obtained with Holistic optimization is tabulated in Table A.47. The prediction performance comparison between the traditional structures and the flexible structure is tabulated in Table A.48. From the 5-tree traditional structure, the 5-tree flexible structure improve prediction performance equivalent to 95.58% of the improvement achieved with the 6-tree traditional structure. This demonstrates that flexible structure was significantly beneficial for prediction performance with the optimization scenario of learning_rate, max_depth, subsample and colsample_bytree for this dataset.

In regards to the structure's characteristics, the traditional structures were relatively similar in values across all hyperparameters. Learning_rate values were roughly 0.51 and

0.46, max_depth was 10 for both structures, the subsample values were roughly 1.0 and 0.97, while colsample were roughly 0.92 and 0.97. The flexible structure had learning_rate values ranging from 0.28 to 0.97. The learning_rate values were relatively similar for Tree 1, 2 and 4, between 0.4 and 0.47. Max_depth values ranged from 8 to 15 with somewhat descending values with later trees. Subsample values were all between 0.94 and 1.0. Colsample_bytree values ranged from 0.8 and 0.97, with all trees except Tree 4 being under 0.9.

	5 Trees	6 Trees
MAE	147.64	146.96
learning_rate	0.5134	0.4638
max_depth	10	10
subsample	0.9996	0.9748
colsample_bytree	0.9177	0.9657

Table A.46: The MAE score and hyperparameter configuration of the traditionally structured ensembles of 5 and 6 trees, based on hyperparameter Scenario 4. Learning_rate, max_depth, subsample and colsample_bytree were optimized though 1000 iterations of Bayesian Optimization, and evaluated with cross validation on the Seoul Bike Sharing dataset.

	learning_rate	max_depth	subsample	colsample_bytree
Tree 1	0.4048	15	0.9996	0.8859
Tree 2	0.4127	16	0.9443	0.8022
Tree 3	0.2819	13	0.9506	0.8436
Tree 4	0.4652	12	0.9849	0.9622
Tree 5	0.9652	8	0.9589	0.8831
MAE	146.99			

Table A.47: The MAE score and hyperparameter configuration of a flexible ensemble structure of 5 trees, based on hyperparameter Scenario 4. The learning_rate, max_depth, subsample and colsample_bytree values for each tree were optimized through 2000 iterations of Bayesian Optimization, and evaluated with cross validation on the Seoul Bike Sharing dataset.

MAE improvement with added tree	0.68
MAE improvement with flexible structure	0.65
Relative percentage improvement	95.58%

Table A.48: The MAE improvement, compared to the traditional structure of 5 trees, with adding another tree to the traditional structure, and with the 5 tree flexible ensemble structure, based on hyperparameter Scenario 4 on the Seoul Bike Sharing dataset. And the percentage of improvement obtained with the flexible structure, relative to that obtained with the added tree.

A.5 Car Evaluation

This section contains the results of the comparison between traditional and flexible structures, each optimized with Bayesian Optimization, on the Car Evaluation dataset. The results are divided into four hyperparmeter scenarios; Scenario 1: learning_rate; Scenario 2: max_depth; Scenario 3: learning_rate, max_depth and subsample; and Scenario 4: learning_rate, max_depth, subsample and colsample_bytree.

Of all the scenarios, only the 5-tree flexible structure of Scenario 1 improved prediction performance compared to the 5-tree traditional structure. The percentage of improvement from the 5-tree traditional structure, relative to the improvement gained with the 6-tree traditional structure, was here 46.60%. For Scenario 2, 3 and 4, the relative percentages of improvement were -38.47%, -146.68% and -7.10%, respectively. These scenarios were thus detrimental to prediction performance compared to the traditional approach to ensemble structure optimization. Scenario 1, being the only scenario with an improvement in prediction with the flexible structure, was also the scenario with the best flexible structure Error, being 0.015339. The best Error of the traditional structures of 5 trees, was comparably Error 0.014181, from Scenario 3. Thus, the flexible ensemble structures could not surpass the best obtained prediction performance of the traditional structures for this dataset.

Regarding the characteristics of the flexible structure, there were some similarities between the scenarios' optimal hyperparameter values. For instance, learning_rate values were generally quite high for all scenarios, and max_depth was very commonly 6, the highest possible value, for most trees. Beyond this, however, the differences between the scenarios were considerable, and it was quite apparent that the hyperparameters influenced each other's optimal values when optimized together.

A.5.1 Scenario 1

For Scenario 1 of the Car Evaluation dataset, the traditionally optimized structures of 5 and 6 trees are tabulated in Table A.49, while the flexible structure obtained with Holistic optimization is tabulated in Table A.50. The prediction performance comparison between the traditional structures and the flexible structure is tabulated in Table A.51. From the 5-tree traditional structure, the 5-tree flexible structure improve prediction performance equivalent to 46.60% of the improvement achieved with the 6-tree traditional structure. This demonstrates that the flexible structure was significantly beneficial for prediction performance with the optimization scenario of learning_rate for this dataset.

In regards to the structure's characteristics, the traditional structures had very similar learning_rate values, both being roughly 0.93. The flexible structure also had quite similar and high learning_rate values, ranging between 0.91 and 1.0.

	5 Trees	6 Trees
Error	0.017362	0.013021
Learning_rate	0.9315	0.9270

Table A.49: The Error score and hyperparameter configuration of the traditionally structured ensembles of 5 and 6 trees, based on hyperparameter Scenario 1. The learning_rate was optimized though 1000 iterations of Bayesian Optimization, and evaluated with cross validation on the Car Evaluation dataset.

Error	0.015339
l_r Tree 1	0.9187
l_r Tree 2	0.9777
l_r Tree 3	0.9979
l_r Tree 4	0.9816
l_r Tree 5	0.9489

Table A.50: The Error score and hyperparameter configuration of a flexible ensemble structure of 5 trees, based on hyperparameter Scenario 1. The learning_rate values for each tree were optimized through 2000 iterations of Bayesian Optimization, and evaluated with cross validation on the Car Evaluation dataset.

Error improvement with added tree	0,004341
Error improvement with flexible structure	0,002023
Relative percentage gain	46.60%

Table A.51: The Error improvement, compared to the traditional structure of 5 trees, with adding another tree to the traditional structure, and with the 5 tree flexible ensemble structure, based on hyperparameter Scenario 1 on the Car Evaluation dataset. And the percentage of improvement obtained with the flexible structure, relative to that obtained with the added tree.

A.5.2 Scenario 2

For Scenario 2 on the Car Evaluation dataset, the traditionally optimized structures of 5 and 6 trees are tabulated in Table A.52, while the flexible structure obtained with Holistic optimization is tabulated in Table A.53. The prediction performance comparison between the traditional structures and the flexible structure is tabulated in Table A.54. From the 5-tree traditional structure, the 5-tree flexible structure improved prediction performance equivalent to -38.47% of the improvement achieved with the 6-tree traditional structure. This demonstrates that the flexible structure was not beneficial for prediction performance with the optimization scenario of max_depth for this dataset.

In regards to the structure's characteristics, the traditional structures were both configured with the max_depth value, 6. For the flexible structure, the max_depth values were 4 and 5 for Tree 1 and 5, respectively, and 6 for the others.

Considering the value range of max_depth being only 1 to 6, the search complexity should be relatively low. It is therefore quite strange how Bayesian Optimization was not able to obtain at least an equal prediction performance to the traditional structure, by finding the equivalent configuration.

	5 Trees	6 Trees
Error	0.052659	0.048898
Max_depth	6	6

Table A.52: The Error score and hyperparameter configuration of the traditionally structured ensembles of 5 and 6 trees, based on hyperparameter Scenario 2. The max_depth was optimized though 1000 iterations of Bayesian Optimization, and evaluated with cross validation on the Car Evaluation dataset.

Error	0.054106
m_d Tree 1	4
m_d Tree 2	6
m_d Tree 3	6
m_d Tree 4	6
m_d Tree 5	5

Table A.53: The Error score and hyperparameter configuration of a flexible ensemble structure of 5 trees, based on hyperparameter Scenario 2. The max_depth values for each tree were optimized through 2000 iterations of Bayesian Optimization, and evaluated with cross validation on the Car Evaluation dataset.

Error improvement with added tree	0.003761
Error improvement with flexible structure	-0.001447
Relative percentage gain	-38.47%

Table A.54: The Error improvement, compared to the traditional structure of 5 trees, with adding another tree to the traditional structure, and with the 5 tree flexible ensemble structure, based on hyperparameter Scenario 2 on the Car Evaluation dataset. And the percentage of improvement obtained with the flexible structure, relative to that obtained with the added tree.

A.5.3 Scenario 3

For Scenario 3 on the Car Evaluation dataset, the traditionally optimized structures of 5 and 6 trees are tabulated in Table A.55, while the flexible structure obtained with Holistic optimization is tabulated in Table A.56. The prediction performance comparison between the traditional structures and the flexible structure is tabulated in Table A.57. From the 5-tree traditional structure, the 5-tree flexible structure improve prediction performance equivalent to -146.68% of the improvement achieved with the 6-tree traditional structure. This demonstrates that flexible structure was detrimental for prediction performance with the optimization scenario of learning_rate, max_depth and subsample for this dataset.

In regards to the structure's characteristics, the traditional structures were nearly identical in all hyperparameter values, with learning_rate being roughly 1.0, max_depth being 6, and subsample being roughly 0.99 for both structures. The flexible structure had learning_rate values ranging between 0.75 and 0.95, the max_depth values were 6 for all trees, while the subsample values ranged between 0.68 and 0.98. However, Tree 5 was the only tree with a subsample value less than 0.89.

	5 Trees	6 Trees
Error	0.014181	0.009841
learning_rate	0.9992	0.9998
max_depth	6	6
subsample	0.9896	0.9872

Table A.55: The Error score and hyperparameter configuration of the traditionally structured ensembles of 5 and 6 trees, based on hyperparameter Scenario 3. Learning_rate, max_depth and subsample were optimized though 1000 iterations of Bayesian Optimization, and evaluated with cross validation on the Car Evaluation dataset.

	learning_rate	max_depth	subsample
Tree 1	0.7554	6	0.9588
Tree 2	0.8824	6	0.8956
Tree 3	0.8270	6	0.9800
Tree 4	0.7974	6	0.9322
Tree 5	0.9475	6	0.6818
Error	0.020547		

Table A.56: The Error score and hyperparameter configuration of a flexible ensemble structure of 5 trees, based on hyperparameter Scenario 3. The learning_rate, max_depth and subsample values for each tree were optimized through 2000 iterations of Bayesian Optimization, and evaluated with cross validation on the Car Evaluation dataset.

Error improvement with added tree	0.00434
Error improvement with flexible structure	-0.006366
Relative percentage gain	-146.68%

Table A.57: The Error improvement, compared to the traditional structure of 5 trees, with adding another tree to the traditional structure, and with the 5 tree flexible ensemble structure, based on hyperparameter Scenario 3 on the Car Evaluation dataset. And the percentage of improvement obtained with the flexible structure, relative to that obtained with the added tree.

A.5.4 Scenario 4

For Scenario 4 on the Car Evaluation dataset, the traditionally optimized structures of 5 and 6 trees are tabulated in Table A.58, while the flexible structure obtained with Holistic optimization is tabulated in Table A.59. The prediction performance comparison between the traditional structures and the flexible structure is tabulated in Table A.60. From the 5-tree traditional structure, the 5-tree flexible structure improve prediction performance equivalent to -7.10% of the improvement achieved with the 6-tree traditional structure. This demonstrates that flexible structure was slightly detrimental to prediction performance with the optimization scenario of learning_rate, max_depth, subsample and colsample_bytree for this dataset.

In regards to the structure's characteristics, the traditional structures were quite similar in most hyperparameter values. The learning_rate values were roughly 1.0 and 0.96, the

max_depth values were 6 for both structures, the subsample values were roughly 0.75 and 0.77, while the colsample_bytree values were roughly 0.90 and 0.91. The flexible structure had learning_rate values ranging between 0.82 and 0.95. The max_depth values were 6 for all trees except the first, where it was 5. Subsample values ranged between 0.68 and 0.88, with only the values of Tree 3 and 5 being lower than 0.85. Colsample_bytree values ranged between 0.81 and 0.99.

	5 Trees	6 Trees
Error	0.028938	0.024884
learning_rate	0.9985	0.9640
max_depth	6	6
subsample	0.7498	0.7676
colsample_bytree	0.9039	0.9058

Table A.58: The Error score and hyperparameter configuration of the traditionally structured ensembles of 5 and 6 trees, based on hyperparameter Scenario 4. Learning_rate, max_depth, subsample and colsample_bytree were optimized though 1000 iterations of Bayesian Optimization, and evaluated with cross validation on the Car Evaluation dataset.

	learning_rate	max_depth	subsample	colsample_bytree
Tree 1	0.8270	5	0.8686	0.8244
Tree 2	0.9429	6	0.8554	0.8137
Tree 3	0.8677	6	0.6852	0.9122
Tree 4	0.9270	6	0.8726	0.9832
Tree 5	0.9003	6	0.7260	0.8512
Error	0.029226			

Table A.59: The Error score and hyperparameter configuration of a flexible ensemble structure of 5 trees, based on hyperparameter Scenario 4. The learning_rate, max_depth, subsample and colsample_bytree values for each tree were optimized through 2000 iterations of Bayesian Optimization, and evaluated with cross validation on the Car Evaluation dataset.

Error improvement with added tree	0.004054
Error improvement with flexible structure	-0.000288
Relative percentage gain	-7.10%

Table A.60: The Error improvement, compared to the traditional structure of 5 trees, with adding another tree to the traditional structure, and with the 5 tree flexible ensemble structure, based on hyperparameter Scenario 4 on the Car Evaluation dataset. And the percentage of improvement obtained with the flexible structure, relative to that obtained with the added tree.

A.6 Statlog Satellite

This section contains the results of the comparison between traditional and flexible structures, each optimized with Bayesian Optimization, on the Statlog Satellite dataset. The results

are divided into four hyperparmeter scenarios; Scenario 1: learning_rate; Scenario 2: max_depth; Scenario 3: learning_rate, max_depth and subsample; and Scenario 4: learning_rate, max_depth, subsample and colsample_bytree.

For all scenarios, the 5-tree flexible structure significantly improved prediction performance compared to the 5-tree traditional structure. The percentage of improvement from the 5-tree traditional structure, relative to the improvement gained with the 6-tree traditional structure, was 86.96%, 24.14%, 91.63% and 88.91%, for Scenario 1 through 4, respectively. Scenario 3 thus had the largest relative percentage of improvement. However, Scenario 4 produced the best Error, being 0.091297. The best Error of the 5-tree traditional structures was comparably 0.092540, from the same Scenario.

Regarding the characteristics of the flexible structure, there were some similarities between the scenarios' optimal values. For instance, for both Scenario 1 and 3 had learning_rate values that somewhat appeared to rise with later trees. The same was observed to be the case for the max_depth values of Scenario 2. In Scenario 3 and 4, max_depth values were observed to exclusively be close to either the lower or higher end of their respective value ranges; 14 to 33 and 12 to 36. Beyond this, however, the differences between the scenarios were considerable, and it was quite apparent that the hyperparameters influenced each other's optimal values when optimized together.

A.6.1 Scenario 1

For Scenario 1 of the Statlog Satellite dataset, the traditionally optimized structures of 5 and 6 trees are tabulated in Table A.61, while the flexible structure obtained with Holistic optimization is tabulated in Table A.62. The prediction performance comparison between the traditional structures and the flexible structure is tabulated in Table A.63. From the 5-tree traditional structure, the 5-tree flexible structure improve prediction performance equivalent to 86.96% of the improvement achieved with the 6-tree traditional structure. This demonstrates that the flexible structure was significantly beneficial for prediction performance with the optimization scenario of learning_rate for this dataset.

In regards to the structure's characteristics, the traditional structures had quite similar learning_rate values, being roughly 0.91 and 0.92 for the 5-tree and 6-tree structure, respectively. The flexible structure had values ranging between 0.34 and 1.0, somewhat appearing to rise with later trees.

	5 Trees	6 Trees
Error	0.102641	0.099067
Learning_rate	0.9135	0.9164

Table A.61: The Error score and hyperparameter configuration of the traditionally structured ensembles of 5 and 6 trees, based on hyperparameter Scenario 1. The learning_rate was optimized though 1000 iterations of Bayesian Optimization, and evaluated with cross validation on the Statlog Satellite dataset.

Error	0.099533
l_r Tree 1	0.3498
l_r Tree 2	0.6041
l_r Tree 3	0.9401
l_r Tree 4	0.9019
l_r Tree 5	0.9998

Table A.62: The Error score and hyperparameter configuration of a flexible ensemble structure of 5 trees, based on hyperparameter Scenario 1. The learning_rate values for each tree were optimized through 2000 iterations of Bayesian Optimization, and evaluated with cross validation on the Statlog Satellite dataset.

Error improvement with added tree	0.003574
Error improvement with flexible structure	0.003108
Relative percentage gain	86.96%

Table A.63: The Error improvement, compared to the traditional structure of 5 trees, with adding another tree to the traditional structure, and with the 5 tree flexible ensemble structure, based on hyperparameter Scenario 1 on the Statlog Satellite dataset. And the percentage of improvement obtained with the flexible structure, relative to that obtained with the added tree.

A.6.2 Scenario 2

For Scenario 2 on the Statlog Satellite dataset, the traditionally optimized structures of 5 and 6 trees are tabulated in Table A.64, while the flexible structure obtained with Holistic optimization is tabulated in Table A.65. The prediction performance comparison between the traditional structures and the flexible structure is tabulated in Table A.66. From the 5-tree traditional structure, the 5-tree flexible structure improved prediction performance equivalent to 24.14% of the improvement achieved with the 6-tree traditional structure. This demonstrates that the flexible structure was significantly beneficial for prediction performance with the optimization scenario of max_depth for this dataset.

In regards to the structure's characteristics, the traditional structures were both configured with the max_depth value, 13. For the flexible structure, the max_depth values ranged from 9 to 28, somewhat appearing to rise with later trees.

	5 Trees	6 Trees
Error	0.098989	0.096736
Max_depth	13	13

Table A.64: The Error score and hyperparameter configuration of the traditionally structured ensembles of 5 and 6 trees, based on hyperparameter Scenario 2. The max_depth was optimized though 1000 iterations of Bayesian Optimization, and evaluated with cross validation on the Statlog Satellite dataset.

Error	0.098445
m_d Tree 1	9
m_d Tree 2	12
m_d Tree 3	26
m_d Tree 4	21
m_d Tree 5	28

Table A.65: The Error score and hyperparameter configuration of a flexible ensemble structure of 5 trees, based on hyperparameter Scenario 2. The max_depth values for each tree were optimized through 2000 iterations of Bayesian Optimization, and evaluated with cross validation on the Statlog Satellite dataset.

Error improvement with added tree	0.002253
Error improvement with flexible structure	0.000544
Relative percentage gain	24.14%

Table A.66: The Error improvement, compared to the traditional structure of 5 trees, with adding another tree to the traditional structure, and with the 5 tree flexible ensemble structure, based on hyperparameter Scenario 2 on the Statlog Satellite dataset. And the percentage of improvement obtained with the flexible structure, relative to that obtained with the added tree.

A.6.3 Scenario 3

For Scenario 3 on the Statlog Satellite dataset, the traditionally optimized structures of 5 and 6 trees are tabulated in Table A.67, while the flexible structure obtained with Holistic optimization is tabulated in Table A.68. The prediction performance comparison between the traditional structures and the flexible structure is tabulated in Table A.69. From the 5-tree traditional structure, the 5-tree flexible structure improve prediction performance equivalent to 91.63% of the improvement achieved with the 6-tree traditional structure. This demonstrates that flexible structure was significantly beneficial for prediction performance with the optimization scenario of learning_rate, max_depth and subsample for this dataset.

In regards to the structure's characteristics, the traditional structures were relatively different in hyperparameter values, with learning_rate being roughly 0.68 and 0.61, max_depth being 36 and 24, and subsample being roughly 0.99 and 0.94. The flexible structure had learning_rate values ranging between 0.43 and 0.90, with values rising with later trees. The max_depth values ranged from 14 to 33, with Tree 2, 3 and 5 being close to the lower value, while the remaining two were close to the higher. Subsample values ranged between 0.77 and 0.97.

	5 Trees	6 Trees
Error	0.095027	0.092229
learning_rate	0.6756	0.6105
max_depth	36	24
subsample	0.9886	0.9408

Table A.67: The Error score and hyperparameter configuration of the traditionally structured ensembles of 5 and 6 trees, based on hyperparameter Scenario 3. Learning_rate, max_depth and subsample were optimized though 1000 iterations of Bayesian Optimization, and evaluated with cross validation on the Statlog Satellite dataset.

	learning_rate	max_depth	subsample
Tree 1	0.4393	33	0.7711
Tree 2	0.7400	14	0.9511
Tree 3	0.8196	15	0.8655
Tree 4	0.8476	32	0.9657
Tree 5	0.8914	14	0.8345
Error	0.092463		

Table A.68: The Error score and hyperparameter configuration of a flexible ensemble structure of 5 trees, based on hyperparameter Scenario 3. The learning_rate, max_depth and subsample values for each tree were optimized through 2000 iterations of Bayesian Optimization, and evaluated with cross validation on the Statlog Satellite dataset.

Error improvement with added tree	0.002798
Error improvement with flexible structure	0.002564
Relative percentage gain	91.63%

Table A.69: The Error improvement, compared to the traditional structure of 5 trees, with adding another tree to the traditional structure, and with the 5 tree flexible ensemble structure, based on hyperparameter Scenario 3 on the Statlog Satellite dataset. And the percentage of improvement obtained with the flexible structure, relative to that obtained with the added tree.

A.6.4 Scenario 4

For Scenario 4 on the Statlog Satellite dataset, the traditionally optimized structures of 5 and 6 trees are tabulated in Table A.70, while the flexible structure obtained with Holistic optimization is tabulated in Table A.71. The prediction performance comparison between the traditional structures and the flexible structure is tabulated in Table A.72. From the 5-tree traditional structure, the 5-tree flexible structure improve prediction performance equivalent to 88.91% of the improvement achieved with the 6-tree traditional structure. This demonstrates that flexible structure was significantly beneficial for prediction performance with the optimization scenario of learning_rate, max_depth, subsample and colsample_bytree for this dataset.

In regards to the structure's characteristics, the traditional structures were relatively different in hyperparameter values. The learning_rate values were roughly 0.58 and 0.65,

the max_depth values were 18 and 28, the subsample values were both roughly 0.98, while colsample_bytree values were roughly 0.84 and 0.88. The flexible structure had learning_rate values ranging between 0.36 and 0.91. Max_depth values ranged from 12 to 36, with Tree 1 and 3 being close to the lower value, and the rest close to the higher. The subsample values ranged between 0.83 and 0.97. The colsample_bytree values ranged between 0.82 and 0.95.

	5 Trees	6 Trees
Error	0.092540	0.091142
learning_rate	0.5802	0.6491
max_depth	18	28
subsample	0.9758	0.9760
colsample_bytree	0.8419	0.8823

Table A.70: The Error score and hyperparameter configuration of the traditionally structured ensembles of 5 and 6 trees, based on hyperparameter Scenario 4. Learning_rate, max_depth, subsample and colsample_bytree were optimized though 1000 iterations of Bayesian Optimization, and evaluated with cross validation on the Statlog Satellite dataset.

	learning_rate	max_depth	subsample	colsample_bytree
Tree 1	0.3676	12	0.8885	0.9379
Tree 2	0.5305	36	0.8707	0.8236
Tree 3	0.9014	13	0.8309	0.9491
Tree 4	0.8453	36	0.9694	0.9262
Tree 5	0.7685	33	0.9464	0.8501
Error	0.091297			

Table A.71: The Error score and hyperparameter configuration of a flexible ensemble structure of 5 trees, based on hyperparameter Scenario 4. The learning_rate, max_depth, subsample and colsample_bytree values for each tree were optimized through 2000 iterations of Bayesian Optimization, and evaluated with cross validation on the Statlog Satellite dataset.

Error improvement with added tree	0.001398
Error improvement with flexible structure	0.001243
Relative percentage gain	88.91%

Table A.72: The Error improvement, compared to the traditional structure of 5 trees, with adding another tree to the traditional structure, and with the 5 tree flexible ensemble structure, based on hyperparameter Scenario 4 on the Statlog Satellite dataset. And the percentage of improvement obtained with the flexible structure, relative to that obtained with the added tree.

A.7 Winequality-red

This section contains the results of the comparison between traditional and flexible structures, each optimized with Bayesian Optimization, on the Winequality-red dataset. The results

are divided into four hyperparmeter scenarios; Scenario 1: learning_rate; Scenario 2: max_depth; Scenario 3: learning_rate, max_depth and subsample; and Scenario 4: learning_rate, max_depth, subsample and colsample_bytree.

For all scenarios, the 5-tree flexible structure improved prediction performance compared to the 5-tree traditional structure. The percentage of improvement from the 5-tree traditional structure, relative to the improvement gained with the 6-tree traditional structure, was 30.90%, 175.56%, 31.31% and 39.91%, for Scenario 1 through 4, respectively. Scenario 2 thus had the largest relative percentage of improvement. However, Scenario 4 produced the best Error, being 0.323957. The best MAE of the 5-tree traditional structures was comparably 0.325826, from the same scenario.

Regarding the characteristics of the flexible structure, there were some similarities between the scenarios' optimal values. For instance most trees of Scenario 2, 3 and 4 had max_depth values ranging from 8 to 11, and Scenario 3 and 4 both had a noticeably lower subsample value for Tree 3. Beyond this, however, the differences between the scenarios were considerable, and it was quite apparent that the hyperparameters influenced each other's optimal values when optimized together.

A.7.1 Scenario 1

For Scenario 1 of the Winequality-red dataset, the traditionally optimized structures of 5 and 6 trees are tabulated in Table A.73, while the flexible structure obtained with Holistic optimization is tabulated in Table A.74. The prediction performance comparison between the traditional structures and the flexible structure is tabulated in Table A.75. From the 5-tree traditional structure, the 5-tree flexible structure improve prediction performance equivalent to 30.90% of the improvement achieved with the 6-tree traditional structure. This demonstrates that the flexible structure was significantly beneficial for prediction performance with the optimization scenario of learning_rate for this dataset.

In regards to the structure's characteristics, the traditional structures had relatively different learning_rate values, being roughly 0.94 and 0.85 for the 5-tree and 6-tree structure, respectively. The flexible structure had values ranging between 0.40 and 0.91, somewhat appearing to rise with later trees.

	5 Trees	6 Trees
Error	0.346769	0.337714
Learning_rate	0.9356	0.8504

Table A.73: The Error score and hyperparameter configuration of the traditionally structured ensembles of 5 and 6 trees, based on hyperparameter Scenario 1. The learning_rate was optimized though 1000 iterations of Bayesian Optimization, and evaluated with cross validation on the Winequality-red dataset.

Error	0.343971
l_r Tree 1	0.4042
l_r Tree 2	0.6531
l_r Tree 3	0.6745
l_r Tree 4	0.9063
l_r Tree 5	0.7769

Table A.74: The Error score and hyperparameter configuration of a flexible ensemble structure of 5 trees, based on hyperparameter Scenario 1. The learning_rate values for each tree were optimized through 2000 iterations of Bayesian Optimization, and evaluated with cross validation on the Winequality-red dataset.

Error improvement with added tree	0.009055
Error improvement with flexible structure	0.002798
Relative percentage gain	30.90%

Table A.75: The Error improvement, compared to the traditional structure of 5 trees, with adding another tree to the traditional structure, and with the 5 tree flexible ensemble structure, based on hyperparameter Scenario 1 on the Winequality-red dataset. And the percentage of improvement obtained with the flexible structure, relative to that obtained with the added tree.

A.7.2 Scenario 2

For Scenario 2 on the Winequality-red dataset, the traditionally optimized structures of 5 and 6 trees are tabulated in Table A.76, while the flexible structure obtained with Holistic optimization is tabulated in Table A.77. The prediction performance comparison between the traditional structures and the flexible structure is tabulated in Table A.78. From the 5-tree traditional structure, the 5-tree flexible structure improved prediction performance equivalent to 175.56% of the improvement achieved with the 6-tree traditional structure. This demonstrates that the flexible structure was very beneficial for prediction performance with the optimization scenario of max_depth for this dataset.

In regards to the structure's characteristics, the traditional structures were both configured with the max_depth value, 11. For the flexible structure, the max_depth values ranged between 3 and 11. However, only Tree 4 had a value less than 9.

	5 Trees	6 Trees
Error	0.336783	0.334283
Max_depth	11	11

Table A.76: The Error score and hyperparameter configuration of the traditionally structured ensembles of 5 and 6 trees, based on hyperparameter Scenario 2. The max_depth was optimized though 1000 iterations of Bayesian Optimization, and evaluated with cross validation on the Winequality-red dataset.

Error	0.332394
m_d Tree 1	10
m_d Tree 2	11
m_d Tree 3	9
m_d Tree 4	3
m_d Tree 5	11

Table A.77: The Error score and hyperparameter configuration of a flexible ensemble structure of 5 trees, based on hyperparameter Scenario 2. The max_depth values for each tree were optimized through 2000 iterations of Bayesian Optimization, and evaluated with cross validation on the Winequality-red dataset.

Error improvement with added tree	0.0025
Error improvement with flexible structure	0.004389
Relative percentage gain	175.56%

Table A.78: The Error improvement, compared to the traditional structure of 5 trees, with adding another tree to the traditional structure, and with the 5 tree flexible ensemble structure, based on hyperparameter Scenario 2 on the Winequality-red dataset. And the percentage of improvement obtained with the flexible structure, relative to that obtained with the added tree.

A.7.3 Scenario 3

For Scenario 3 on the Winequality-red dataset, the traditionally optimized structures of 5 and 6 trees are tabulated in Table A.79, while the flexible structure obtained with Holistic optimization is tabulated in Table A.80. The prediction performance comparison between the traditional structures and the flexible structure is tabulated in Table A.81. From the 5-tree traditional structure, the 5-tree flexible structure improve prediction performance equivalent to 31.31% of the improvement achieved with the 6-tree traditional structure. This demonstrates that flexible structure was significantly beneficial for prediction performance with the optimization scenario of learning_rate, max_depth and subsample for this dataset.

In regards to the structure's characteristics, the traditional structures were slightly different in hyperparameter values, with learning_rate being roughly 0.64 and 0.59, max_depth being 10 for both structures, and subsample being roughly 0.92 and 0.6. The flexible structure had learning_rate values ranging between 0.57 and 0.97. The max_depth values ranged from 8 to 11. And subsample values ranged between 0.67 and 0.89, though only Tree 3 had a value less than 0.77.

	5 Trees	6 Trees
Error	0.327406	0.317392
learning_rate	0.6438	0.5881
max_depth	10	10
subsample	0.9176	0.9578

Table A.79: The Error score and hyperparameter configuration of the traditionally structured ensembles of 5 and 6 trees, based on hyperparameter Scenario 3. Learning_rate, max_depth and subsample were optimized though 1000 iterations of Bayesian Optimization, and evaluated with cross validation on the Winequality-red dataset.

	learning_rate	max_depth	subsample
Tree 1	0.5794	8	0.8880
Tree 2	0.7289	11	0.8182
Tree 3	0.9633	10	0.6719
Tree 4	0.6776	8	0.8362
Tree 5	0.7411	9	0.7760
Error	0.32427		

Table A.80: The Error score and hyperparameter configuration of a flexible ensemble structure of 5 trees, based on hyperparameter Scenario 3. The learning_rate, max_depth and subsample values for each tree were optimized through 2000 iterations of Bayesian Optimization, and evaluated with cross validation on the Winequality-red dataset.

Error improvement with added tree	0.010014
Error improvement with flexible structure	0.003136
Relative percentage gain	31.31%

Table A.81: The Error improvement, compared to the traditional structure of 5 trees, with adding another tree to the traditional structure, and with the 5 tree flexible ensemble structure, based on hyperparameter Scenario 3 on the Winequality-red dataset. And the percentage of improvement obtained with the flexible structure, relative to that obtained with the added tree.

A.7.4 Scenario 4

For Scenario 4 on the Winequality-red dataset, the traditionally optimized structures of 5 and 6 trees are tabulated in Table A.82, while the flexible structure obtained with Holistic optimization is tabulated in Table A.83. The prediction performance comparison between the traditional structures and the flexible structure is tabulated in Table A.84. From the 5-tree traditional structure, the 5-tree flexible structure improve prediction performance equivalent to 39.91% of the improvement achieved with the 6-tree traditional structure. This demonstrates that flexible structure was significantly beneficial for prediction performance with the optimization scenario of learning_rate, max_depth, subsample and colsample_bytree for this dataset.

In regards to the structure's characteristics, the traditional structures were relatively different in hyperparameter values. The learning_rate values were roughly 0.91 and 0.65,

the max_depth values were 11 and 9, the subsample values were roughly 0.94 and 0.96, while the colsample_bytree values were roughly 0.91 and 0.93. The flexible structure had learning_rate values ranging between 0.58 and 0.75. Max_depth values were all relatively high, ranging between 8 and 11. The subsample values ranged between 0.67 and 0.97, but only Tree 3 had a value less than 0.82. Colsample_bytree values ranged between 0.81 and 0.98.

	5 Trees	6 Trees
Error	0.325826	0.321144
learning_rate	0.9125	0.6495
max_depth	11	9
subsample	0.9377	0.9626
colsample_bytree	0.9134	0.9323

Table A.82: The Error score and hyperparameter configuration of the traditionally structured ensembles of 5 and 6 trees, based on hyperparameter Scenario 4. Learning_rate, max_depth, subsample and colsample_bytree were optimized though 1000 iterations of Bayesian Optimization, and evaluated with cross validation on the Winequality-red dataset.

	learning_rate	max_depth	subsample	colsample_bytree
Tree 1	0.5899	9	0.9289	0.8424
Tree 2	0.7496	9	0.8400	0.9526
Tree 3	0.6835	8	0.6703	0.9539
Tree 4	0.6946	10	0.8264	0.8128
Tree 5	0.7484	11	0.9667	0.9769
Error	0.323957			

Table A.83: The Error score and hyperparameter configuration of a flexible ensemble structure of 5 trees, based on hyperparameter Scenario 4. The learning_rate, max_depth, subsample and colsample_bytree values for each tree were optimized through 2000 iterations of Bayesian Optimization, and evaluated with cross validation on the Winequality-red dataset.

Error improvement with added tree	0.004682
Error improvement with flexible structure	0.001869
Relative percentage gain	39.91%

Table A.84: The Error improvement, compared to the traditional structure of 5 trees, with adding another tree to the traditional structure, and with the 5 tree flexible ensemble structure, based on hyperparameter Scenario 4 on the Winequality-red dataset. And the percentage of improvement obtained with the flexible structure, relative to that obtained with the added tree.

A.8 Hold-out Test-set

To ensure that the obtained MAEs were not overfitted to the cross validation folds, we conducted an additional optimization process with a hold-out test-set. For this process,

we selected Energy Prediction as the dataset, as this is the largest of the selected datasets. The ensemble was optimized with Energy Prediction's dedicated training-set through the same process as described in 5.1. After the final ensemble configuration was obtained, its cross validation score was documented, it was trained on the training-set and evaluated on the test-set, held out during optimization and training.

The obtained flexible ensemble configuration, along with its cross validation- and test MAE, is tabulated in Table A.85. Comparing the cross validation MAE and the test MAE, we can see that the test MAE was actually quite a bit better than that of the cross validation. This demonstrates that overfitting to the cross validation folds is likely not an issue.

	learning_rate	max_depth	subsample	colsample_bytree
Tree 1	0.1574	25	0.8785	0.8848
Tree 2	0.5622	3	0.7534	0.9606
Tree 3	0.2927	27	0.9725	0.8790
Tree 4	0.4942	27	0.9739	0.9557
Tree 5	0.4738	22	0.9748	0.9747
CV MAE	34.2314			
Test MAE	31.6840			

Table A.85: A flexible ensemble structure, based on hyperparameter Scenario 4, optimized based on cross validation, and evaluated on a hold-out test set on the Energy Prediction dataset. The learning_rate, max_depth, subsample and colsample_bytree values for each tree were optimized through 2000 iterations of Bayesian Optimization.

Appendix B

Experiment 2 Per-Dataset Results

B.1 Concrete

The results of Experiment 1-2 on the Concrete dataset are tabulated in Table B.1. We can see that the first tree obtained an MAE of roughly 6.72, and was gradually improved with the following trees until the final MAE of roughly 4.67, was reached with Tree 5. We can, however, see that the improvement in prediction performance decreased with each added tree.

In regards to to the learning_rate values, all were very high, ranging between 0.86 and 1.0. However, besides Tree 5, all trees had values higher than 0.96.

Tree	MAE	Selected learning_rate
Tree 1	6.7243	0.9997
Tree 2	5.4630	0.9670
Tree 3	5.0550	0.9891
Tree 4	4.7785	0.9970
Tree 5	4.6703	0.8650

Table B.1: The flexible ensemble structure obtained with the Incremental optimization approach for the Concrete dataset. The MAE score and selected learning_rate values for each tree, optimized through 200 iterations of Bayesian Optimization, are included.

B.2 Energy Prediction

The results of Experiment 1-2 on the Energy Prediction dataset are tabulated in Table B.2. We can see that the first tree obtained an MAE of 45.14, and was gradually improved with the following trees until the final MAE of roughly 44.18 was reached with Tree 5. We can, however, see that the improvement in prediction performance significantly decreased with each added tree. For instance, Tree 4 and 5 were nearly identical in prediction performance.

Regarding the learning_rate values, these ranged between 0.05 and 0.76. However, besides Tree 1, all trees had values less than 0.27, decreasing with each tree. The fact that the learning_rate values quickly became so low, could be an indicator that the prediction performance has converged or gotten stuck in a local optima.

Tree	MAE	Selected learning_rate
Tree 1	45.1400	0.7633
Tree 2	44.4539	0.2625
Tree 3	44.2411	0.1708
Tree 4	44.1956	0.0849
Tree 5	44.1813	0.0539

Table B.2: The flexible ensemble structure obtained with the Incremental optimization approach for the Energy Prediction dataset. The MAE score and selected learning_rate value for each tree, optimized through 200 iterations of Bayesian Optimization, are included.

B.3 Housing

The results of Experiment 1-2 on the Housing dataset are tabulated in Table B.3. We can see that the first tree obtained an MAE of roughly 3.4, and was gradually improved with the following trees until the final MAE of roughly 2.87 was reached with Tree 5. We can, however, see that the improvement in prediction performance significantly decreased with each added tree.

Regarding the learning_rate values, these ranged between roughly 0.48 and 1.0, beginning with the higher value and decreasing with later added trees.

Tree	MAE	Selected learning_rate
Tree 1	3.4046	0.9999
Tree 2	3.0257	0.9220
Tree 3	2.9388	0.6546
Tree 4	2.9043	0.4905
Tree 5	2.8731	0.4804

Table B.3: The flexible ensemble structure obtained with the Incremental optimization approach for the Housing dataset. The MAE score and selected learning_rate value for each tree, optimized through 200 iterations of Bayesian Optimization, are included.

B.4 Seoul Bike Sharing

The results of Experiment 1-2 on the Seoul Bike Sharing dataset are tabulated in Table B.4. We can see that first tree obtained an MAE of roughly 208, and was gradually improved with the following trees until the final MAE of roughly 171, was reached with the Tree 5. We can, however, see that the improvement in prediction performance decreased with each added tree.

In regards to the learning_rate values, all of the values were quite high, ranging between 0.62 and 0.97. However, besides Tree 5, all trees had values higher than 0.78. The values appeared to be gradually decreasing from the higher value with later added trees.

Tree	MAE	Selected learning_rate
Tree 1	208.34	0.9690
Tree 2	186.14	0.8542
Tree 3	179.42	0.7885
Tree 4	174.29	0.8043
Tree 5	171.55	0.6278

Table B.4: The flexible ensemble structure obtained with the Incremental optimization approach for the Seoul Bike Sharing dataset. The MAE score and selected learning_rate value for each tree, optimized through 200 iterations of Bayesian Optimization, are included.

B.5 Car Evaluation

The results of Experiment 1-2 on the Car Evaluation dataset are tabulated in Table B.5. We can see that the first two trees obtained an Error of roughly 0.0440, and was gradually improved with the following trees until the final Error of roughly 0.0191 was reached with Tree 5. We can, however, see that the improvement in prediction performance decreased with each added tree

Regarding the learning_rate values, these were all quite high, ranging between 0.82 and 1.0. This is further exemplified by the fact that Tree 4 and 5 were the only ones with values less than 0.98. Overall, the values somewhat appeared to be decreasing for each added tree.

Tree	Error	Selected learning_rate
Tree 1	-	0.9943
Tree 2	0.043974	0.9949
Tree 3	0.033272	0.9886
Tree 4	0.026040	0.8403
Tree 5	0.019101	0.8215

Table B.5: The flexible ensemble structure obtained with the Incremental optimization approach for the Car Evaluation dataset. Each tree was optimized through 200 iterations of Bayesian Optimization, except Tree 1 and 2, which were optimized together through 400 iterations. The Error and selected learning_rate values are included for each optimized tree, except for Tree 1, where the Error score was inaccessible.

B.6 Statlog Satellite

The results of Experiment 1-2 on the Statlog Satellite dataset are tabulated in Table B.6. We can see that the first two trees obtained an Error of roughly 0.1155, and was gradually improved with the following trees until the final Error of roughly 0.1024 was reached with Tree 5.

Regarding the learning_rate values, these were all relatively high, ranging between 0.74 and 1.0. However, only tree 1 and 3 had values less than 0.96.

Tree	Error	Selected learning_rate
Tree 1	-	0.7413
Tree 2	0.115462	0.9986
Tree 3	0.110800	0.8689
Tree 4	0.105905	0.9856
Tree 5	0.102408	0.9616

Table B.6: The flexible ensemble structure obtained with the Incremental optimization approach for the Statlog Satellite dataset. Each tree was optimized through 200 iterations of Bayesian Optimization, except Tree 1 and 2, which were optimized together through 400 iterations. The Error and selected learning_rate values are included for each optimized tree, except for Tree 1, where the Error score was inaccessible.

B.7 Winequality-red

The results of Experiment 1-2 on the Winequality-red dataset are tabulated in Table B.7. We can see that the first two trees obtained an Error of roughly 0.3771, and was gradually improved with the following trees until the final Error of roughly 0.3562 was reached with Tree 5.

Regarding the learning_rate values, these ranged between 0.58 and 0.98, and somewhat appeared to be rising with later added trees.

Tree	Error	Selected learning_rate
Tree 1	-	0.5842
Tree 2	0.377115	0.7139
Tree 3	0.369919	0.9412
Tree 4	0.360537	0.8153
Tree 5	0.356157	0.9709

Table B.7: The flexible ensemble structure obtained with the Incremental optimization approach for the Winequality-red dataset. Each tree was optimized through 200 iterations of Bayesian Optimization, except Tree 1 and 2, which were optimized together through 400 iterations. The Error and selected learning_rate values are included for each optimized tree, except for Tree 1, where the Error score was inaccessible.

Appendix C

Experiment 3 Per-Dataset Results

C.1 Concrete

This section contains the results of Experiment 2-1 on the Concrete dataset. The results are divided into two hyperparameter scenarios; Scenario 1: learning_rate optimized in isolation; and Scenario 2: learning_rate and max_depth optimized together.

In both Scenarios, the average, best and worst prediction performance were better for the flexible ensemble structures, with large differences in the cases of average and worst, and only slight in the best case. The reason for the large differences in the average and worst cases can be explained from the histograms. The histograms of the flexible structures were more concentrated in a range of better performance, compared to the traditional structures where a larger portion of values were of worse prediction performance. Overall, it was clear that it was easier to obtain better prediction performance with flexible structures for both hyperparameter scenarios on the dataset of Concrete.

C.1.1 Scenario 1

The comparison between the average, best and worst prediction performance of traditional and flexible ensemble structures of Scenario 1 are tabulated in Table C.1, and the prediction performance histograms of each structure are contained in Figure C.1.

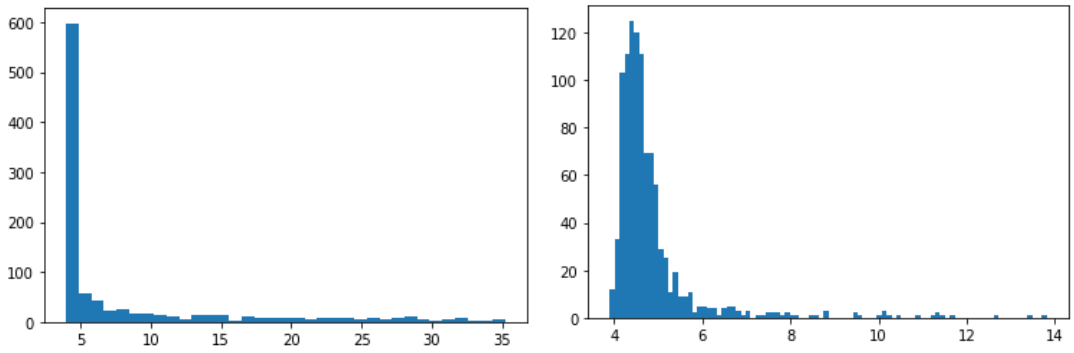
The prediction performance of the flexible structures were better in all cases of the average, best and worst values. The differences were especially significant in the average and worst cases, while it was only slight in the best case.

The histograms were somewhat similar, but the values of the flexible structures seemed to be more concentrated in the 4 to 6 MAE value range. In the traditional structure histograms, many values were similarly located within this range, but there was also a significant portion of worse values. This is in turn the reason for the better values of the flexible structures in the average and worst cases.

Overall, it was quite clear that it was easier to obtain better prediction performance with with the flexible ensemble structures for Scenario 1 on Concrete, despite the higher search complexity.

Concrete Scenario 1			
Approach	Average	Best	Worst
Traditional	8.3171	4.0149	35.1962
Flexible	4.8377	3.9042	13.8173
Difference	3.4797	0.1107	21.3789

Table C.1: The average, best and worst cross validation score (MAE) from Scenario 1 of Experiment 2-1 on the Concrete dataset. The value differences between the traditional and Flexible structure optimization approaches are also included. Positive differences indicate that Flexible was better, while the opposite for negative differences.



(a) Traditional (b) Flexible

Figure C.1: Histograms of cross validation scores from Scenario 1 of Experiment 2-1 on the Concrete dataset. Figure (a) represents the cross validation scores obtained with the Traditional ensemble structure optimization approach, while Figure (b) represents the ones obtained with the Flexible approach. The horizontal axis of the histograms indicate the score values (MAE), and the vertical axis indicate the number of times these values occurred.

C.1.2 Scenario 2

The comparison between the average, best and worst prediction performance of traditional and flexible ensemble structures of Scenario 2 are tabulated in Table C.2, and the prediction performance histograms of each structure are contained in Figure C.2.

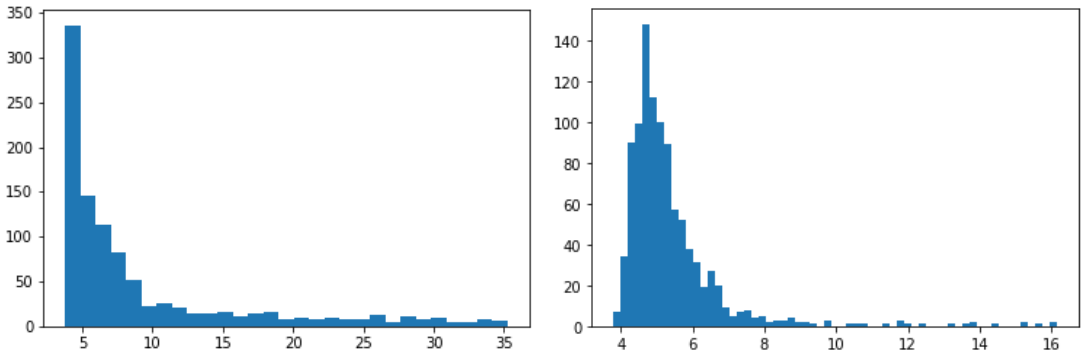
The prediction performance of the flexible structures were better in all cases of the average, best and worst values. The differences were especially significant in the average and worst cases, while it was only slight in the best case.

The histograms were relatively similar, but the values of the flexible structures seemed to be more concentrated in the 4 to 7 MAE value range. In the traditional structure histograms, many values were similarly located within this range, but there was also a significant portion of worse values. This is in turn the reason for the better values of the flexible structures in the average and worst cases.

Overall, it was quite clear that it was easier to obtain better prediction performance with with the flexible ensemble structures for Scenario 2 on Concrete, despite the higher search complexity.

Concrete Scenario 2			
Approach	Average	Best	Worst
Traditional	9.2206	3.8162	35.2361
Flexible	5.4038	3.8034	16.1442
Difference	3.8168	0.0128	19.0919

Table C.2: The average, best and worst cross validation score (MAE) from Scenario 2 of the "General Insight" investigation on the Concrete dataset. The value differences between the traditional and Flexible structure optimization approaches are also included. Positive differences indicate that Flexible was better, while the opposite for negative differences.



(a) Traditional (b) Flexible

Figure C.2: Histograms of cross validation scores from Scenario 2 of Experiment 2-1 on the Concrete dataset. Figure (a) represents the cross validation scores obtained with the Traditional ensemble structure optimization approach, while Figure (b) represents the ones obtained with the Flexible approach. The horizontal axis of the histograms indicate the score values (MAE), and the vertical axis indicate the number of times these values occurred.

C.2 Energy Prediction

This section contains the results of Experiment 2-1 on the Energy Prediction dataset. The results are divided into two hyperparameter scenarios; Scenario 1: learning_rate optimized in isolation; and Scenario 2: learning_rate and max_depth optimized together.

In both Scenarios, the average and worst prediction performance were better for the flexible ensemble structures, while the best prediction performance was only better for the flexible structures in Scenario 2. The differences were typically larger in the cases of average and worst, and smaller in the best value case. The reason for the large differences in the average and worst cases can be explained from the histograms. The histograms of the flexible structures were more concentrated in a range of better performance, compared to the

traditional structures where a larger portion of values were of worse prediction performance. Overall, it was clear that it was easier to obtain better prediction performance with flexible structures for Scenario 2, while they were competitive with traditional structures for Scenario 1, on the dataset of Energy Prediction.

C.2.1 Scenario 1

The comparison between the average, best and worst prediction performance of traditional and flexible ensemble structures of Scenario 1 are tabulated in Table C.3, and the prediction performance histograms of each structure are contained in Figure C.3.

The prediction performance of the flexible structures were better in the cases of the average and worst values, but worse in the best values case. The differences were especially significant in the worst cases, while smaller in the average and best case.

The histograms were relatively similar, but the values of the flexible structures seemed to be considerably more concentrated in the 42 to 48 MAE range. In the traditional structure histograms, most values were similarly located within this range, but there was also a significant portion of worse values. This explains the better values of the flexible structures in the average and worst cases.

Overall, it seemed like the flexible ensemble structures made search difficulty in Scenario 1 easier by making it harder to obtain worse values, while being somewhat competitive in best obtained prediction performance.

Energy Prediction Scenario 1			
Approach	Average	Best	Worst
Traditional	48.7335	41.8257	97.1359
Flexible	45.9559	42.2148	51.8404
Difference	2.7776	-0.3891	45.2955

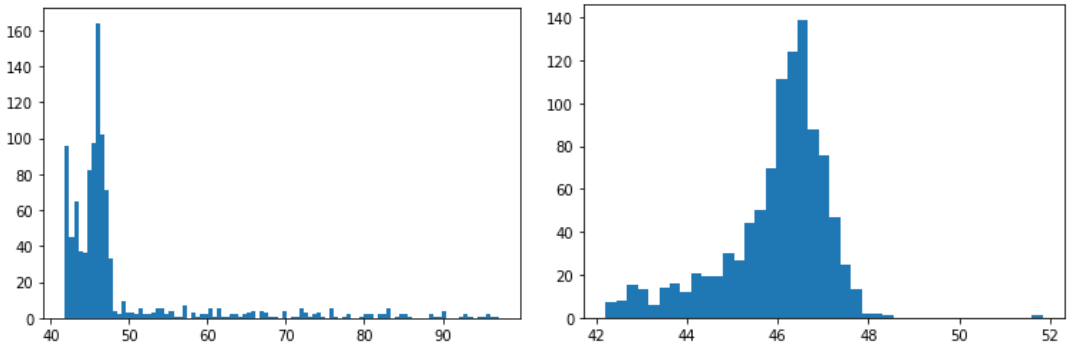
Table C.3: The average, best and worst cross validation score from Scenario 1 of the "General Insight" investigation on the Energy Prediction dataset. The value differences between the traditional and Flexible structure optimization approaches are also included. Positive differences indicate that Flexible was better, while the opposite for negative differences.

C.2.2 Scenario 2

The comparison between the average, best and worst prediction performance of traditional and flexible ensemble structures of Scenario 2 are tabulated in Table C.4, and the prediction performance histograms of each structure are contained in Figure C.4.

The prediction performance of the flexible structures were better in all cases of the average, best and worst values. The differences were especially significant in the average and worst cases, while smaller in the best case.

The histograms were quite different for this scenario. The values of the flexible structures were concentrated in the 33 to 42 MAE range. In the traditional structure histograms, while many values were located within this range, there was also a large amount of worse values. This explains the better values of the flexible structures in the average and worst cases.



(a) Traditional (b) Flexible

Figure C.3: Histograms of cross validation scores from Scenario 1 of Experiment 2-1 on the Energy Prediction dataset. Figure (a) represents the cross validation scores obtained with the Traditional ensemble structure optimization approach, while Figure (b) represents the ones obtained with the Flexible approach. The horizontal axis of the histograms indicate the score values (MAE), and the vertical axis indicate the number of times these values occurred.

Overall, it was quite clear that it was easier to obtain better prediction performance with with the flexible ensemble structures for Scenario 2 on Energy Prediction, despite the higher search complexity.

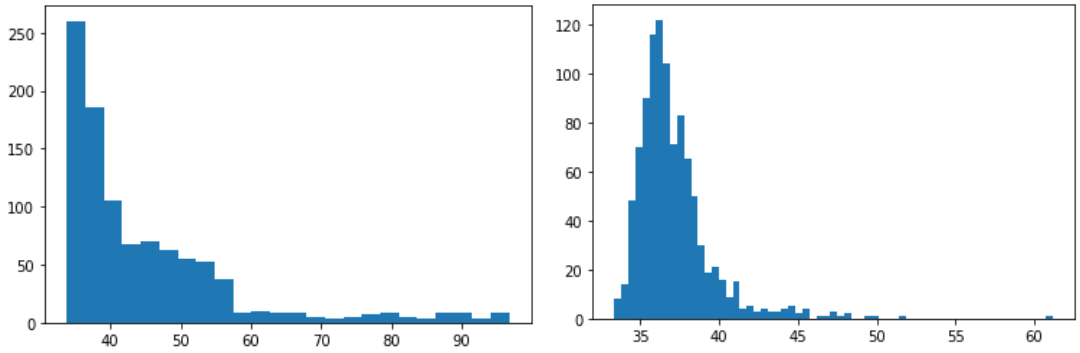
Energy Prediction Scenario 2			
Approach	Average	Best	Worst
Traditional	45,133	33,8593	96,7704
Flexible	37,1433	33,3772	61,1627
Difference	7.9897	0.4821	35.6077

Table C.4: The average, best and worst cross validation score from Scenario 2 of the "General Insight" investigation on the Energy Prediction dataset. The value differences between the traditional and Flexible structure optimization approaches are also included. Positive differences indicate that Flexible was better, while the opposite for negative differences.

C.3 Housing

This section contains the results of Experiment 2-1 on the Housing dataset. The results are divided into two hyperparameter scenarios; Scenario 1: learning_rate optimized in isolation; and Scenario 2: learning_rate and max_depth optimized together.

In both Scenarios, the average, best and worst prediction performance were better for the flexible ensemble structures, with large differences in the cases of average and worst, and only slight in the best case. The reason for the large differences in the average and worst cases can be explained from the histograms. The histograms of the flexible structures were more concentrated in a range of better performance, compared to the traditional structures where a larger portion of values were of worse prediction performance. Overall, it was clear



(a) Traditional (b) Flexible

Figure C.4: Histograms of cross validation scores from Scenario 2 of Experiment 2-1 on the Energy Prediction dataset. Figure (a) represents the cross validation scores obtained with the Traditional ensemble structure optimization approach, while Figure (b) represents the ones obtained with the Flexible approach. The horizontal axis of the histograms indicate the score values (MAE), and the vertical axis indicate the number of times these values occurred.

that it was easier to obtain better prediction performance with flexible structures for both hyperparameter scenarios on the dataset of Housing.

C.3.1 Scenario 1

The comparison between the average, best and worst prediction performance of traditional and flexible ensemble structures of Scenario 1 are tabulated in Table C.5, and the prediction performance histograms of each structure are contained in Figure C.5.

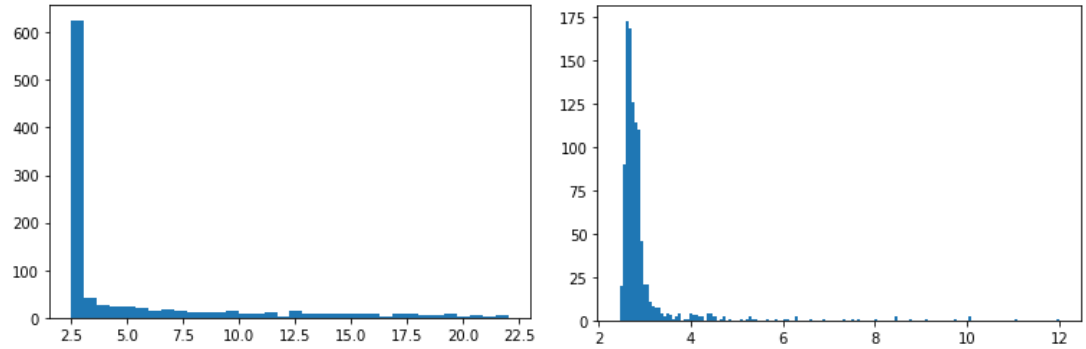
The prediction performance of the flexible structures were better in all cases of the average, best and worst values. The differences were especially significant in the average and worst cases, while much smaller in the best case.

The histograms were relatively similar, but the values of the flexible structures seemed to be considerably more concentrated in the 2 to 4 MAE range. In the traditional structure histograms, many values were similarly located within this range, but there was also a significant portion of worse values. This explains the better values of the flexible structures in the average and worst cases.

Overall, it was quite clear that it was easier to obtain better prediction performance with with the flexible ensemble structures for Scenario 1 on Housing, despite the higher search complexity.

Housing Scenario 1			
Approach	Average	Best	Worst
Traditional	5.1684	2.4964	22.0247
Flexible	2.9258	2.4475	11.998
Difference	2.2426	0.0489	10.0267

Table C.5: The average, best and worst cross validation score from Scenario 1 of the "General Insight" investigation on the Housing dataset. The value differences between the traditional and Flexible structure optimization approaches are also included. Positive differences indicate that Flexible was better, while the opposite for negative differences.



(a) Traditional (b) Flexible

Figure C.5: Histograms of cross validation scores from Scenario 1 of Experiment 2-1 on the Housing dataset. Figure (a) represents the cross validation scores obtained with the Traditional ensemble structure optimization approach, while Figure (b) represents the ones obtained with the Flexible approach. The horizontal axis of the histograms indicate the score values (MAE), and the vertical axis indicate the number of times these values occurred.

C.3.2 Scenario 2

The comparison between the average, best and worst prediction performance of traditional and flexible ensemble structures of Scenario 2 are tabulated in Table C.6, and the prediction performance histograms of each structure are contained in Figure C.6.

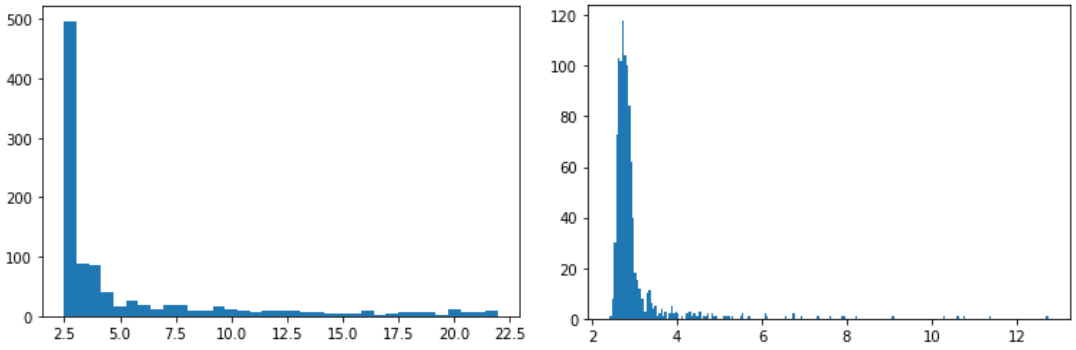
The prediction performance of the flexible structures were better in all cases of the average, best and worst values. The differences were especially significant in the average and worst cases, while much smaller in the best case.

The histograms were relatively similar, but the values of the flexible structures seemed to be more concentrated in the 2 to 4 MAE range. In the traditional structure histograms, many values were similarly located within this range, but there was also a significant portion of worse values. This explains the better values of the flexible structures in the average and worst cases.

Overall, it was quite clear that it was easier to obtain better prediction performance with with the flexible ensemble structures for Scenario 2 on Housing, despite the higher search complexity.

Housing Scenario 2			
Approach	Average	Best	Worst
Traditional	5.434	2.4937	21.9435
Flexible	2.9825	2.4198	12.7407
Difference	2.4515	0.0739	9.2028

Table C.6: The average, best and worst cross validation score from Scenario 2 of the "General Insight" investigation on the Housing dataset. The value differences between the traditional and Flexible structure optimization approaches are also included. Positive differences indicate that Flexible was better, while the opposite for negative differences.



(a) Traditional

(b) Flexible

Figure C.6: Histograms of cross validation scores from Scenario 2 of Experiment 2-1 on the Housing dataset. Figure (a) represents the cross validation scores obtained with the Traditional ensemble structure optimization approach, while Figure (b) represents the ones obtained with the Flexible approach. The horizontal axis of the histograms indicate the score values (MAE), and the vertical axis indicate the number of times these values occurred.

C.4 Seoul Bike Sharing

This section contains the results of Experiment 2-1 on the Seoul Bike Sharing dataset. The results are divided into two hyperparameter scenarios; Scenario 1: learning_rate optimized in isolation; and Scenario 2: learning_rate and max_depth optimized together.

In both Scenarios, the average and worst prediction performance were better for the flexible ensemble structures, while the best prediction performance was only better for the flexible structures in Scenario 1. The differences were typically larger in the cases of average and worst. This can be explained from the histograms. The histograms of the flexible structures were more concentrated in a range of better performance, compared to the traditional structures where a larger portion of values were of worse prediction performance.

Overall, it was clear that it was easier to obtain better prediction performance with flexible structures for Scenario 1, while it was a bit more mixed in Scenario 2, on the dataset of Seoul Bike Sharing.

C.4.1 Scenario 1

The comparison between the average, best and worst prediction performance of traditional and flexible ensemble structures of Scenario 1 are tabulated in Table C.7, and the prediction performance histograms of each structure are contained in Figure C.7.

The prediction performance of the flexible structures were better in all cases of the average, best and worst values. The differences were especially significant in the average and worst cases, while much smaller in the best case.

The histograms were relatively similar, but the values of the flexible structures seemed to be considerably more concentrated in the 150 to 200 MAE range. In the traditional structure histograms, many values were similarly located within this range, but there was also a large amount of worse values. This explains the better values of the flexible structures in the average and worst cases.

Overall, it was quite clear that it was easier to obtain better prediction performance with with the flexible ensemble structures for Scenario 1 on Seoul Bike Sharing, despite the higher search complexity.

Seoul Bike Sharing Scenario 1			
Approach	Average	Best	Worst
Traditional	240.85	162.15	704.05
Flexible	174.25	161.36	361.69
Difference	66.6	0.79	342.36

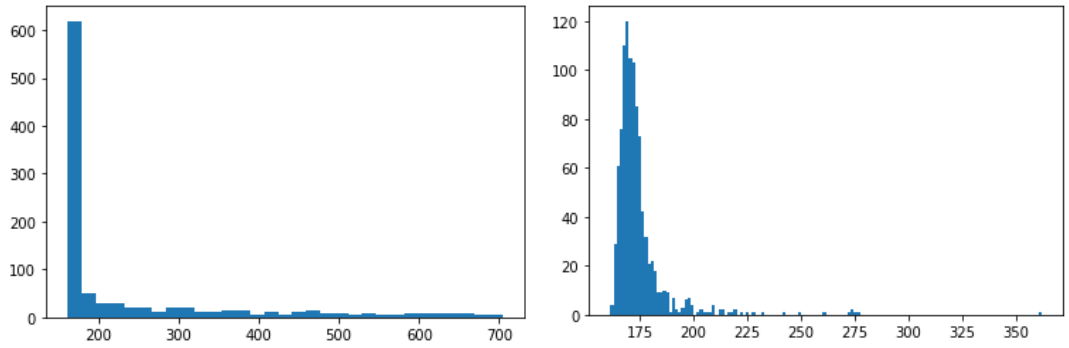
Table C.7: The average, best and worst cross validation score from Scenario 1 of the "General Insight" investigation on the Seoul Bike Sharing dataset. The value differences between the traditional and Flexible structure optimization approaches are also included. Positive differences indicate that Flexible was better, while the opposite for negative differences.

C.4.2 Scenario 2

The comparison between the average, best and worst prediction performance of traditional and flexible ensemble structures of Scenario 2 are tabulated in Table C.8, and the prediction performance histograms of each structure are contained in Figure C.8.

The prediction performance of the flexible structures were better in the cases of the average and worst values, but worse in the best values case. The differences were quite significant in all cases.

The histograms were relatively similar, but the values of the flexible structures seemed to be considerably more concentrated in the 250 to 300 MAE range. In the traditional structure histograms, many values were similarly located within this range, but there was also a large amount of worse values. This explains the better values of the flexible structures in the average and worst cases.



(a) Traditional (b) Flexible

Figure C.7: Histograms of cross validation scores from Scenario 1 of Experiment 2-1 on the Seoul Bike Sharing dataset. Figure (a) represents the cross validation scores obtained with the Traditional ensemble structure optimization approach, while Figure (b) represents the ones obtained with the Flexible approach. The horizontal axis of the histograms indicate the score values (MAE), and the vertical axis indicate the number of times these values occurred.

Overall, it seemed like the flexible ensemble structures made search difficulty in Scenario 2 easier by making it harder to obtain worse values, while not quite managing to compete with traditional structure in terms of best prediction performance.

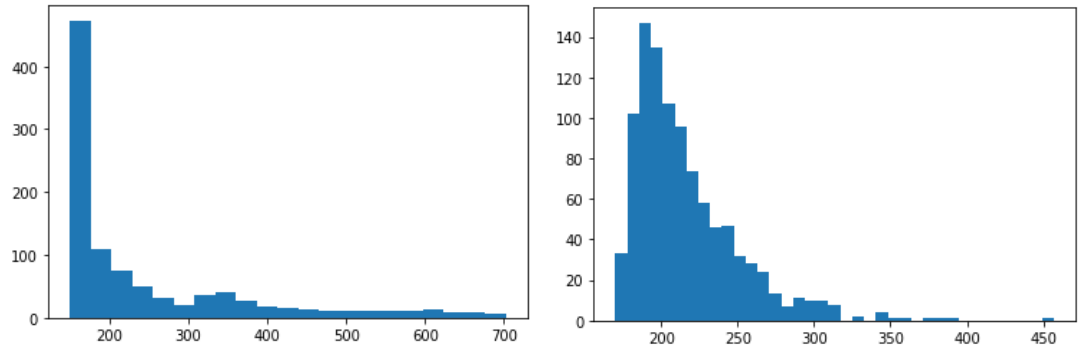
Seoul Bike Sharing Scenario 2			
Approach	Average	Best	Worst
Traditional	248.17	150.28	702.2
Flexible	216.19	169.98	456.73
Difference	31.98	-19.70	245.47

Table C.8: The average, best and worst cross validation score from Scenario 2 of the "General Insight" investigation on the Seoul Bike Sharing dataset. The value differences between the traditional and Flexible structure optimization approaches are also included. Positive differences indicate that Flexible was better, while the opposite for negative differences.

C.5 Car Evaluation

This section contains the results of Experiment 2-1 on the Car Evaluation dataset. The results are divided into two hyperparameter scenarios; Scenario 1: learning_rate optimized in isolation; and Scenario 2: learning_rate and max_depth optimized together.

In both scenarios, the average and worst prediction performance were better for the flexible ensemble structures, while the best prediction performance was significantly worse. The histograms of Scenario 1 were quite similar, while they were relatively different in Scenario 2. Overall, while the flexible structures were better on average for both scenarios, it seemed more difficult to find as good prediction performance as traditional structures in the best value case.



(a) Traditional (b) Flexible

Figure C.8: Histograms of cross validation scores from Scenario 2 of Experiment 2-1 on the Seoul Bike Sharing dataset. Figure (a) represents the cross validation scores obtained with the Traditional ensemble structure optimization approach, while Figure (b) represents the ones obtained with the Flexible approach. The horizontal axis of the histograms indicate the score values (MAE), and the vertical axis indicate the number of times these values occurred.

C.5.1 Scenario 1

The comparison between the average, best and worst prediction performance of traditional and flexible ensemble structures of Scenario 1 are tabulated in Table C.9, and the prediction performance histograms of each structure are contained in Figure C.9.

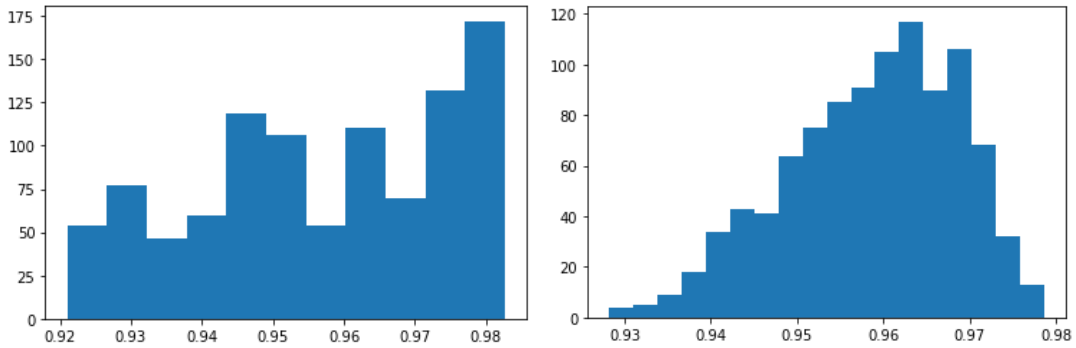
The prediction performance of the flexible structures were better in the cases of the average and worst values, but worse in the best values case. The differences were quite small in all cases.

The histograms were quite similar. Both had values ranging between 92% and 98% Accuracy. The great similarity between the histograms could explain why the differences are only slight.

Overall, it was a bit unclear which structure type was better for Scenario 1 on Car Evaluation.

Car Evaluation Scenario 1			
Approach	Average	Best	Worst
Traditional	95.70%	98.26%	92.10%
Flexible	95.86%	97.85%	92.82%
Difference	0.16%	-0.41%	0.72%

Table C.9: The average, best and worst cross validation score from Scenario 1 of the "General Insight" investigation on the Car Evaluation dataset. The value differences between the traditional and Flexible structure optimization approaches are also included. Positive differences indicate that Flexible was better, while the opposite for negative differences.



(a) Traditional (b) Flexible

Figure C.9: Histograms of cross validation scores from Scenario 1 of Experiment 2-1 on the Car Evaluation dataset. Figure (a) represents the cross validation scores obtained with the Traditional ensemble structure optimization approach, while Figure (b) represents the ones obtained with the Flexible approach. The horizontal axis of the histograms indicate the score values (Accuracy), and the vertical axis indicate the number of times these values occurred.

C.5.2 Scenario 2

The comparison between the average, best and worst prediction performance of traditional and flexible ensemble structures of Scenario 2 are tabulated in Table C.10, and the prediction performance histograms of each structure are contained in Figure C.10.

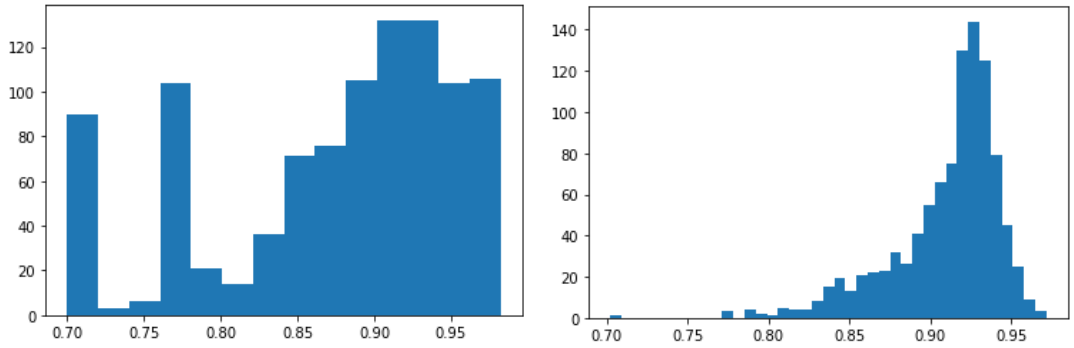
The prediction performance of the flexible structures were better in the cases of the average and worst values, but worse in the best values case. The difference was quite significant in the average and best cases, and much smaller in the worst case.

The histograms were relatively different for this scenario. The values of the flexible structures were concentrated in the 90% to 97% Accuracy range. In the traditional structure histograms, while many values were located within this range, there was also a large amount of worse values. This explains the significantly better values of the flexible structures in the average case.

Overall, it was a bit unclear which structure type was better for Scenario 1 on Car Evaluation. The flexible structures were better on average, but could not reach reach the best prediction performance of the traditional structures.

Car Evaluation Scenario 2			
Approach	Average	Best	Worst
Traditional	87.41%	99.50%	70.02%
Flexible	91.00%	97.16%	70.22%
Difference	3.59%	-2.34%	0.20%

Table C.10: The average, best and worst cross validation score (Accuracy) from Scenario 2 of the "General Insight" investigation on the Car Evaluation dataset. The value differences between the traditional and Flexible structure optimization approaches are also included. Positive differences indicate that Flexible was better, while the opposite for negative differences.



(a) Traditional

(b) Flexible

Figure C.10: Histograms of cross validation scores from Scenario 2 of Experiment 2-1 on the Car Evaluation dataset. Figure (a) represents the cross validation scores obtained with the Traditional ensemble structure optimization approach, while Figure (b) represents the ones obtained with the Flexible approach. The horizontal axis of the histograms indicate the score values (Accuracy), and the vertical axis indicate the number of times these values occurred.

C.6 Statlog Satellite

This section contains the results of Experiment 2-1 on the Statlog Satellite dataset. The results are divided into two hyperparameter scenarios; Scenario 1: learning_rate optimized in isolation; and Scenario 2: learning_rate and max_depth optimized together.

In both Scenarios, the average and best prediction performance of the flexible ensemble structures were roughly equal to that of the traditional structures, while the worst value case was significantly better. This can be explained from the histograms. These were very similar i both scenarios, with only some worse values contained in the traditional structure histograms. Overall it seemed like the flexible and traditional structure types were roughly as effective for the Statlog Satellite dataset, with a slight benefit with flexible structures.

C.6.1 Scenario 1

The comparison between the average, best and worst prediction performance of traditional and flexible ensemble structures of Scenario 1 are tabulated in Table C.11, and the prediction performance histograms of each structure are contained in Figure C.11.

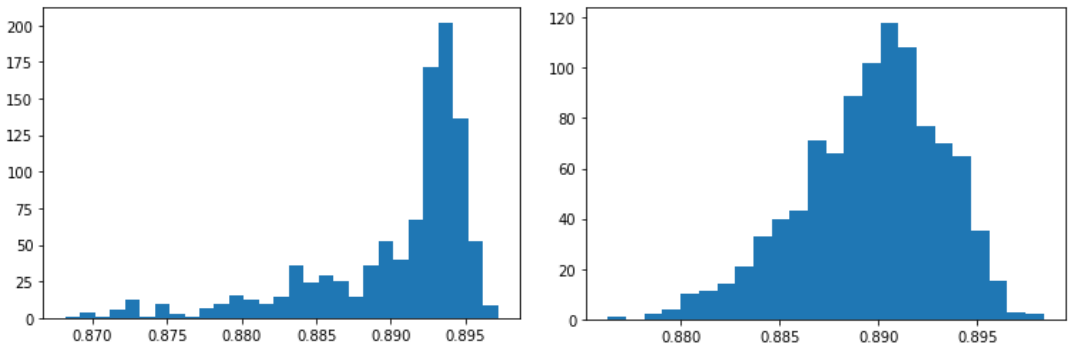
The prediction performance of the flexible structure were better in the cases of best and worst values, and ever so slightly worse in the average case. The differences were quite small in all cases.

The histograms were relatively similar, but the values of the flexible structures seemed to be a bit more concentrated in the 88% to 98.5% Accuracy range. In the traditional structure histograms, many values were similarly located within this range, but there was also a decent amount of worse values. This explains the better values of the flexible structures in the worst case.

Overall, it seemed like it was easier to obtain better prediction performance with the flexible structures for Scenario 1 on Statlog Satellite, despite the higher search complexity.

Statlog Satellite Scenario 1			
Approach	Average	Best	Worst
Traditional	89.03%	89.72%	86.81%
Flexible	88.96%	89.83%	87.63%
Difference	-0.07%	0.11%	0.82%

Table C.11: The average, best and worst cross validation score from Scenario 1 of the "General Insight" investigation on the Statlog Satellite dataset. The value differences between the traditional and Flexible structure optimization approaches are also included. Positive differences indicate that Flexible was better, while the opposite for negative differences.



(a) Traditional

(b) Flexible

Figure C.11: Histograms of cross validation scores from Scenario 1 of Experiment 2-1 on the Statlog Satellite dataset. Figure (a) represents the cross validation scores obtained with the Traditional ensemble structure optimization approach, while Figure (b) represents the ones obtained with the Flexible approach. The horizontal axis of the histograms indicate the score values (Accuracy), and the vertical axis indicate the number of times these values occurred.

C.6.2 Scenario 2

The comparison between the average, best and worst prediction performance of traditional and flexible ensemble structures of Scenario 2 are tabulated in Table C.12, and the prediction performance histograms of each structure are contained in Figure C.12.

The prediction performance of the flexible structures were better in the cases of the average and worst values, but worse in the best values case. The differences were quite small in all cases.

The histograms were relatively similar, but the values of the flexible structures seemed to be a bit more concentrated in the 88% to 90.5% Accuracy range. In the traditional structure histogram, most values were similarilly located within this range, but there were also a small amount of worse values. This explains the better values of the flexible structures in the worst case.

Overall, it seemed like it was slightly easier to obtain better prediction performance with the flexible structures for Scenario 2 on Statlog Satellite, despite the higher search complexity.

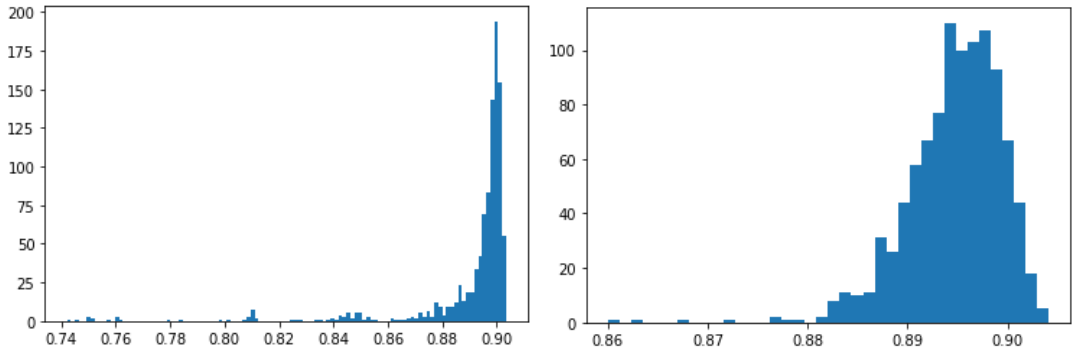
Statlog Satellite Scenario 2			
Approach	Average	Best	Worst
Traditional	89.07%	90.34%	74.20%
Flexible	89.46%	90.40%	86.00%
Difference	0.39%	0.06%	11.80%

Table C.12: The average, best and worst cross validation score from Scenario 2 of the "General Insight" investigation on the Statlog Satellite dataset. The value differences between the traditional and Flexible structure optimization approaches are also included. Positive differences indicate that Flexible was better, while the opposite for negative differences.

C.7 Winequality-red

This section contains the results of Experiment 2-1 on the Winequality-red dataset. The results are divided into two hyperparameter scenarios; Scenario 1: learning_rate optimized in isolation; and Scenario 2: learning_rate and max_depth optimized together.

In Scenario 1, the best and worst value cases were better for the flexible structures, but the average was not. In Scenario 2, the average and worst value cases were better, but not the best value. The differences were for all value cases, except the worst case in Scenario 2, quite small. This can be explained from the histograms. These were quite similar in both scenarios, with the biggest difference being a slightly larger number of worse values for the traditional structures, especially in Scenario 2. Overall it seemed like the flexible and traditional structure types were about equally effective for the Winequality-red dataset, with perhaps a slight benefit with flexible structures.



(a) Traditional (b) Flexible

Figure C.12: Histograms of cross validation scores from Scenario 2 of Experiment 2-1 on the Statlog Satellite dataset. Figure (a) represents the cross validation scores obtained with the Traditional ensemble structure optimization approach, while Figure (b) represents the ones obtained with the Flexible approach. The horizontal axis of the histograms indicate the score values (Accuracy), and the vertical axis indicate the number of times these values occurred.

C.7.1 Scenario 1

The comparison between the average, best and worst prediction performance of traditional and flexible ensemble structures of Scenario 1 are tabulated in Table C.13, and the prediction performance histograms of each structure are contained in Figure C.13.

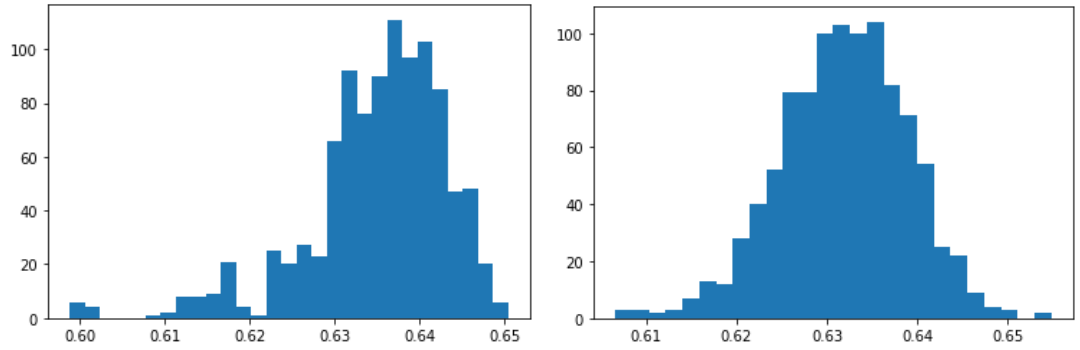
The prediction performance of the flexible structure were better in the cases of best and worst values, and worse in the average case. However, the differences were quite small in all cases.

The histograms were nearly identical, though a bit more even in distributions with the flexible structures. This explains the very small differences in all value cases.

Overall, the two structure types seemed about equally efficient for obtaining prediction performance in Scenario 1 on Winequality-red, with a slight benefit for with the flexible structures.

Winequality-red Scenario 1			
Approach	Average	Best	Worst
Traditional	63.50%	65.04%	59.88%
Flexible	63.18%	65.47%	60.66%
Difference	-0.32%	0.43%	0.78%

Table C.13: The average, best and worst cross validation score from Scenario 1 of the "General Insight" investigation on the Winequality-red dataset. The value differences between the traditional and Flexible structure optimization approaches are also included. Positive differences indicate that Flexible was better, while the opposite for negative differences.



(a) Traditional (b) Flexible

Figure C.13: Histograms of cross validation scores from Scenario 1 of Experiment 2-1 on the Winequality-red dataset. Figure (a) represents the cross validation scores obtained with the Traditional ensemble structure optimization approach, while Figure (b) represents the ones obtained with the Flexible approach. The horizontal axis of the histograms indicate the score values (Accuracy), and the vertical axis indicate the number of times these values occurred.

C.7.2 Scenario 2

The comparison between the average, best and worst prediction performance of traditional and flexible ensemble structures of Scenario 2 are tabulated in Table C.14, and the prediction performance histograms of each structure are contained in Figure C.14.

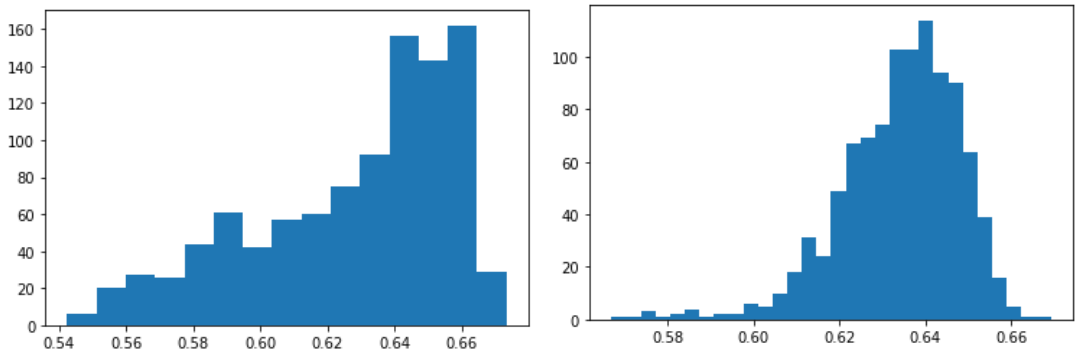
The prediction performance of the flexible structures were better in the cases of the average and worst values, but worse in the best values case. The difference was relatively small in the average and best cases, and a bit larger in the worst value case.

The histograms were relatively similar, but the values of the flexible structures seemed to be a bit more concentrated in the 60% to 66% Accuracy range. In the traditional structure histogram, most values were similarly located within this range, but there were also a considerable amount of worse values. This explains the better values of the flexible structures in the average and worst case.

Overall, it seemed like the flexible ensemble structures made search difficulty in Scenario 2 easier by making it harder to obtain worse values, while not quite managing to compete with traditional structure in terms of best prediction performance.

Winequality-red Scenario 2			
Approach	Average	Best	Worst
Traditional	62.80%	67.32%	54.25%
Flexible	63.41%	66.91%	56.69%
Difference	0.61%	-0.41%	2.44%

Table C.14: The average, best and worst cross validation score from Scenario 2 of the "General Insight" investigation on the Winequality-red dataset. The value differences between the traditional and Flexible structure optimization approaches are also included. Positive differences indicate that Flexible was better, while the opposite for negative differences.



(a) Traditional (b) Flexible

Figure C.14: Histograms of cross validation scores from Scenario 2 of Experiment 2-1 on the Winequality-red dataset. Figure (a) represents the cross validation scores obtained with the Traditional ensemble structure optimization approach, while Figure (b) represents the ones obtained with the Flexible approach. The horizontal axis of the histograms indicate the score values (Accuracy), and the vertical axis indicate the number of times these values occurred.

Appendix D

Experiment 4 Per-Dataset Results

D.1 Concrete

The ten best and worst performing configurations of Experiment 2-2 on the Concrete dataset are tabulated in Table D.1 and D.2, respectively.

The prediction performance of the ten best configurations ranged from 3.98 to 4.25 MAE. Generally, the configurations were quite similar in learning_rate values for the different trees. For instance, the average of the first tree was 0.34, of which most configurations had a relatively close value, derived from the relatively low standard deviation. The same can be said for the other trees. The 10 best configurations were thus similar, but clearly distinguishable.

The prediction performance of the ten worst configurations ranged from 21.88 to 14.95 MAE. Much like with the best configurations, the worst configurations were quite similar. For instance, the average of the first tree was 0.11, of which all configurations had a relatively close value, derived from the low standard deviation. The same can be said for the other trees. The 10 worst configurations were thus similar, but clearly distinguishable.

Comparing the best and worst configurations on the Concrete dataset, it was apparent that the learning_rate values were very different between these cases. This can be demonstrated with the learning_rate averages, which were very low for the worst configurations, compared to those of the best. These differences, as well as the similarities in characteristics between the configurations in each prediction performance group, indicate that the area of best performance in the learning_rate search landscape can be predicted and exploited. The fact that similar prediction performance was achieved with similar, but slightly different configurations, also indicate that the area of best performance is quite lenient. Roughly the avearage plus/minus the standard deviation for each tree.

Prediction Performance	learning_rate				
MAE	Tree 1	Tree 2	Tree 3	Tree 4	Tree 5
3.98	0.2973	0.7243	0.8922	0.6257	0.792
4.16	0.4753	0.3991	0.8599	0.7023	0.7353
4.17	0.4673	0.8628	0.8325	0.9877	0.0102
4.17	0.7087	0.8149	0.788	0.506	0.5611
4.19	0.0873	0.6842	0.8344	0.6221	0.9719
4.21	0.4005	0.6927	0.5399	0.9725	0.7636
4.22	0.257	0.3327	0.5648	0.868	0.7437
4.23	0.219	0.3616	0.9334	0.9681	0.8076
4.24	0.376	0.9865	0.9668	0.7969	0.8295
4.25	0.1908	0.7094	0.9099	0.7913	0.9102
Average:	0.3479	0.6568	0.8122	0.7841	0.7125
Standard Deviation:	0.1777	0.2221	0.1467	0.1678	0.2699

Table D.1: The 10 best flexible structure configurations for the Concrete dataset, based on cross validation scores from Experiment 2-2. The average learning_rate values and standard deviations for each tree are included.

Prediction Performance	learning_rate				
MAE	Tree 1	Tree 2	Tree 3	Tree 4	Tree 5
21.88	0.1846	0.0437	0.0236	0.0956	0.1284
21.28	0.0459	0.0222	0.0784	0.2126	0.1298
19.41	0.0531	0.0055	0.2053	0.0513	0.259
17.8	0.0958	0.1695	0.2603	0.0848	0.0635
16.49	0.0658	0.2047	0.0356	0.0881	0.3252
15.98	0.2031	0.0026	0.0534	0.3403	0.1536
15.98	0.0577	0.0951	0.2947	0.0642	0.233
15.33	0.0173	0.228	0.115	0.3843	0.0229
14.97	0.3938	0.0936	0.1632	0.1045	0.047
14.95	0.0302	0.0936	0.172	0.4109	0.0781
Average:	0.1147	0.0959	0.1402	0.1837	0.1441
Standard Deviation:	0.1163	0.0815	0.0946	0.1422	0.0997

Table D.2: The 10 worst flexible structure configurations for the Concrete dataset, based on cross validation scores from Experiment 2-2. The average learning_rate values and standard deviations for each tree are included.

D.2 Energy Prediction

The ten best and worst performing configurations of Experiment 2-2 on the Energy Prediction dataset are tabulated in Table D.3 and D.4, respectively.

The prediction performance of the ten best configurations ranged from 41.09 to 41.57 MAE. Generally, the configurations were quite similar in learning_rate values for the different trees. For instance, the average of the first tree was 0.21, of which most configurations had a relatively close value, derived from the low standard deviation. The

same can be said for the other trees. The 10 best configurations were thus similar, but clearly distinguishable.

The prediction performance of the ten worst configurations ranged from 81.23 to 50.76 MAE. Much like with the best configurations, the worst configurations were quite similar. For instance, the average of the first tree was 0.10, of which all configurations had a relatively close value, derived from the low standard deviation. The same can be said for the other trees. The 10 worst configurations were thus similar, but clearly distinguishable.

Comparing the best and worst configurations on the Energy Prediction dataset, it was apparent that the learning_rate values were somewhat different between these cases. This can be demonstrated with the learning_rate averages, which were a bit lower for the worst configurations, compared to those of the best. These differences, as well as the similarities in characteristics between the configurations in each prediction performance group, indicate that the area of best performance in the learning_rate search landscape can be predicted and exploited. The fact that similar prediction performance was achieved with similar, but slightly different configurations, also indicate that the area of best performance is relatively lenient. Roughly the average plus/minus the standard deviation for each tree.

Prediction Performance	learning_rate				
MAE	Tree 1	Tree 2	Tree 3	Tree 4	Tree 5
41.0953	0.2322	0.2034	0.2327	0.4338	0.3156
41.1142	0.2258	0.2154	0.1205	0.2789	0.3608
41.3158	0.2366	0.4056	0.0584	0.1460	0.3104
41.3254	0.2313	0.1252	0.3408	0.2681	0.3502
41.4169	0.2119	0.2895	0.1917	0.2848	0.3750
41.4941	0.1214	0.3881	0.2843	0.4308	0.2215
41.5155	0.2789	0.1525	0.3251	0.3299	0.3975
41.5310	0.2364	0.4497	0.3166	0.0980	0.3214
41.5396	0.2431	0.2592	0.3282	0.1782	0.4124
41.5771	0.1332	0.1750	0.3969	0.3866	0.3133
Average:	0.2151	0.2664	0.2595	0.2835	0.3378
Standard Deviation:	0.0494	0.1136	0.1072	0.1163	0.0546

Table D.3: The 10 best flexible structure configurations for the Energy Prediction dataset, based on cross validation scores from Experiment 2-2. The average learning_rate values and standard deviations for each tree are included.

Prediction Performance	learning_rate				
MAE	Tree 1	Tree 2	Tree 3	Tree 4	Tree 5
81.2304	0.0854	0.0055	0.0086	0.0052	0.0563
56.3671	0.3352	0.0104	0.0282	0.0341	0.0739
56.0490	0.1851	0.0073	0.0647	0.0424	0.2165
55.2854	0.0005	0.2789	0.0839	0.1148	0.0442
53.2990	0.0161	0.0537	0.0675	0.2414	0.1973
53.0094	0.0976	0.1212	0.1642	0.0372	0.1739
52.8724	0.0810	0.2834	0.1922	0.0031	0.0111
52.1578	0.0675	0.2530	0.0100	0.0975	0.1661
51.7248	0.1699	0.0040	0.0947	0.3200	0.0039
50.7607	0.0440	0.0822	0.2604	0.1294	0.1256
Average:	0.1082	0.1100	0.0974	0.1025	0.1069
Standard Deviation:	0.0992	0.1182	0.0833	0.1051	0.0788

Table D.4: The 10 worst flexible structure configurations for the Energy Prediction dataset, based on cross validation scores from Experiment 2-2. The average learning_rate values and standard deviations for each tree are included.

D.3 Housing

The ten best and worst performing configurations of Experiment 2-2 on the Housing dataset are tabulated in Table D.5 and D.6, respectively.

The prediction performance of the ten best configurations ranged from 1.93 to 2.04 MAE. Generally, the configurations were relatively similar in learning_rate values for the different trees. For instance, the average of the first tree was 0.21, of which most configurations had a relatively close value, derived from the low standard deviation. The same can be said for the other trees, though Tree 4 seemed a bit more variable, having a standard deviation above 0.3. The 10 best configurations were thus similar, but clearly distinguishable.

The prediction performance of the ten worst configurations ranged from 13 to 9.42 MAE. Much like with the best configurations, the worst configurations were quite similar. For instance, the average of the first tree was 0.11, of which all configurations had a relatively close value, derived from the low standar deviation. The same can be said for the other trees. The 10 worst configurations were thus similar, but clearly distinguishable.

Comparing the best and worst configurations on the Housing dataset, it was apparent that the learning_rate values were very different between these cases. This can be demonstrated with the learning_rate averages, which were very low for the worst configurations, compared to those of the best. These differences, as well as the similarities in characteristics between the configurations in each prediction performance group, indicate that the area of best performance in the learning_rate search landscape can be predicted and exploited. The fact that similar prediction performance was achieved with similar, but slightly different configurations, also indicate that the area of best performance is relatively lenient. Roughly the average plus/minus the standard deviation for each tree.

Prediction Performance	learning_rate				
MAE	Tree 1	Tree 2	Tree 3	Tree 4	Tree 5
1.9375	0.1252	0.8377	0.6303	0.0902	0.9413
1.9637	0.1348	0.8259	0.7035	0.6960	0.5602
1.9678	0.3407	0.1750	0.6587	0.0169	0.9190
1.9991	0.3966	0.4589	0.5189	0.9953	0.2886
2.0042	0.1416	0.7925	0.3292	0.7715	0.9820
2.0341	0.1112	0.1220	0.9747	0.9812	0.5494
2.0376	0.2837	0.5777	0.8343	0.9726	0.5331
2.0445	0.1790	0.8787	0.8990	0.5076	0.8405
2.0447	0.3133	0.8571	0.8111	0.5090	0.6214
2.0462	0.1150	0.4818	0.7448	0.9301	0.8465
Average:	0.2141	0.6007	0.7105	0.6470	0.7082
Standard Deviation:	0.1081	0.2852	0.1895	0.3625	0.2289

Table D.5: The 10 best flexible structure configurations for the Housing dataset, based on cross validation scores from Experiment 2-2. The average learning_rate values and standard deviations for each tree are included.

Prediction Performance	learning_rate				
MAE	Tree 1	Tree 2	Tree 3	Tree 4	Tree 5
13.0099	0.0587	0.0732	0.0534	0.1907	0.0901
11.0470	0.1299	0.1015	0.0659	0.1032	0.2177
10.8734	0.1220	0.2302	0.1029	0.0467	0.1263
10.7568	0.0301	0.0275	0.3376	0.0403	0.1776
10.6067	0.0644	0.0755	0.2351	0.0206	0.2448
10.5536	0.0234	0.3362	0.0854	0.0495	0.1343
10.1349	0.1503	0.3404	0.0563	0.0372	0.0912
10.0187	0.1792	0.3331	0.0173	0.1338	0.0176
9.7780	0.2376	0.0632	0.2909	0.0840	0.0304
9.4237	0.1757	0.1981	0.0390	0.2425	0.0983
Average:	0.1171	0.1779	0.1284	0.0949	0.1228
Standard Deviation:	0.0712	0.1255	0.1150	0.0737	0.0740

Table D.6: The 10 worst flexible structure configurations for the Housing dataset, based on cross validation scores from Experiment 2-2. The average learning_rate values and standard deviations for each tree are included.

D.4 Seoul Bike Sharing

The ten best and worst performing configurations of Experiment 2-2 on the Seoul Bike Sharing dataset are tabulated in Table D.7 and D.8, respectively.

The prediction performance of the ten best configurations ranged from 158.61 to 161.98 MAE. Generally, the configurations were quite similar in learning_rate values for the different trees. For instance, the average of the first tree was 0.46, of which all configurations

had a relatively close value, derived from the low standard deviation. The same can be said for the other trees. The 10 best configurations were thus similar, but clearly distinguishable.

The prediction performance of the ten worst configurations ranged from 447 to 343.81 MAE. Much like with the best configurations, the worst configurations were quite similar. For instance, the average of the first tree was 0.15, of which all configurations had a relatively close value, derived from the low standard deviation. The same can be said for the other trees. The 10 worst configurations were thus similar, but clearly distinguishable.

Comparing the best and worst configurations on the Seoul Bike Sharing dataset, it was apparent that the learning_rate values were very different between these cases. This can be demonstrated with the learning_rate averages, which were very low for the worst configurations, compared to those of the best. These differences, as well as the similarities in characteristics between the configurations in each prediction performance group, indicate that the area of best performance in the learning_rate search landscape can be predicted and exploited. The fact that similar prediction performance was achieved with similar, but slightly different configurations, also indicate that the area of best performance is relatively lenient. Roughly the average plus/minus the standard deviation for each tree.

Prediction Performance	learning_rate				
MAE	Tree 1	Tree 2	Tree 3	Tree 4	Tree 5
158.61	0.5273	0.3989	0.7306	0.9932	0.5576
159.15	0.3732	0.5087	0.8396	0.5465	0.9629
160.44	0.5252	0.4618	0.7617	0.8963	0.9926
160.58	0.5227	0.4553	0.7556	0.8777	0.786
160.66	0.2621	0.5754	0.7738	0.7255	0.8149
160.67	0.4232	0.6331	0.4553	0.6662	0.8628
161.29	0.5623	0.8453	0.7528	0.9199	0.638
161.66	0.4581	0.6855	0.5342	0.6603	0.8314
161.82	0.5379	0.7262	0.6016	0.8429	0.6669
161.98	0.4221	0.7587	0.7062	0.4919	0.61
Average:	0.4614	0.6049	0.6911	0.7620	0.7723
Standard Deviation:	0.0935	0.1486	0.1210	0.1689	0.1491

Table D.7: The 10 best flexible structure configurations for the Seoul Bike Sharing dataset, based on cross validation scores from Experiment 2-2. The average learning_rate values and standard deviations for each tree are included.

Prediction Performance	learning_rate				
MAE	Tree 1	Tree 2	Tree 3	Tree 4	Tree 5
447	0.0736	0.0609	0.0986	0.026	0.1961
397.23	0.0728	0.2722	0.0212	0.0891	0.1101
374.31	0.1434	0.279	0.1273	0.0382	0.0321
373.54	0.1286	0.0077	0.1887	0.0477	0.2521
368.36	0.0576	0.1124	0.2171	0.019	0.2351
364.22	0.1428	0.0212	0.0323	0.2146	0.2399
357.29	0.1555	0.0006	0.1026	0.0407	0.3491
354.18	0.3346	0.1026	0.0857	0.1311	0.0171
347.67	0.3534	0.2176	0.0131	0.0779	0.0133
343.81	0.0989	0.0354	0.255	0.0434	0.2783
Average:	0.1561	0.1110	0.1142	0.0728	0.1723
Standard Deviation:	0.1047	0.1079	0.0836	0.0602	0.1206

Table D.8: The 10 worst flexible structure configurations for the Seoul Bike Sharing dataset, based on cross validation scores from Experiment 2-2. The average learning_rate values and standard deviations for each tree are included.

D.5 Car Evaluation

The ten best and worst performing configurations of Experiment 2-2 on the Car Evaluation dataset are tabulated in Table D.9 and D.10, respectively.

The prediction performance of the ten best configurations were exactly 97.97% Accuracy. Generally, the configurations were quite similar in learning_rate values for the different trees. For instance, the average of the first tree was 0.84, of which most configurations had a relatively close value, derived from the relatively low standard deviation. The same can be said for the other trees. The 10 best configurations were thus similar, but clearly distinguishable.

The prediction performance of the ten worst configurations ranged from 93.06% to 93.93% Accuracy. Much like with the best configurations, the worst configurations were relatively similar. For instance, the average of the first tree was 0.08, of which most configurations had a relatively close value, derived from the low standard deviation. The same can be said for the other trees, though the last two trees seemed a bit more variable. The 10 worst configurations were thus similar, but clearly distinguishable.

Comparing the best and worst configurations on the Car Evaluation dataset, the learning_rate values were quite different based on the first three trees, where the average were considerably lower. These differences, as well as the similarities in characteristics between the configurations in each prediction performance group, indicate that the area of best performance in the learning_rate search landscape can be predicted and exploited. The fact that similar prediction performance was achieved with similar, but slightly different configurations, also indicate that the area of best performance is relatively lenient. Roughly the average plus/minus the standard deviation for each tree.

Prediction Performance	learning_rate				
Accuracy	Tree 1	Tree 2	Tree 3	Tree 4	Tree 5
97.97%	0.5821	0.4967	0.8327	0.5056	0.7843
97.97%	0.8484	0.9923	0.4427	0.7402	0.6889
97.97%	0.4487	0.8058	0.5471	0.6914	0.6060
97.97%	0.7356	0.7264	0.8819	0.9562	0.8213
97.97%	0.7109	0.6868	0.2093	0.8026	0.6677
97.97%	0.5947	0.6604	0.2807	0.4813	0.5160
97.97%	0.2037	0.5690	0.6839	0.6156	0.6069
97.97%	0.8636	0.7014	0.4379	0.7817	0.6151
97.97%	0.8638	0.6787	0.4234	0.7250	0.3839
97.97%	0.8891	0.4379	0.9270	0.5144	0.9304
Average:	0.8498	0.6755	0.5666	0.6814	0.6620
Standard Deviation	0.2212	0.1566	0.2530	0.1523	0.1564

Table D.9: The 10 best flexible structure configurations for the Car Evaluation dataset, based on cross validation scores from Experiment 2-2. The average learning_rate values and standard deviations for each tree are included.

Prediction Performance	learning_rate				
Accuracy	Tree 1	Tree 2	Tree 3	Tree 4	Tree 5
93.06%	0.0088	0.1973	0.1069	0.8142	0.1515
93.06%	0.1869	0.2932	0.3401	0.0824	0.9740
93.06%	0.0105	0.2720	0.1631	0.6809	0.0275
93.06%	0.0312	0.0872	0.1452	0.9860	0.3675
93.35%	0.0017	0.1114	0.2371	0.2634	0.6137
93.64%	0.1430	0.3250	0.2750	0.9479	0.1705
93.64%	0.2714	0.3944	0.0855	0.1484	0.8479
93.93%	0.0371	0.0243	0.0644	0.0442	0.3602
93.93%	0.0166	0.2015	0.0240	0.4470	0.7809
93.93%	0.1048	0.3988	0.3580	0.8624	0.0027
Average:	0.0812	0.2305	0.1799	0.5276	0.4296
Standard Deviation:	0.0924	0.1286	0.1169	0.3736	0.3537

Table D.10: The 10 worst flexible structure configurations for the Car Evaluation dataset, based on cross validation scores from Experiment 2-2. The average learning_rate values and standard deviations for each tree are included.

D.6 Statlog Satellite

The ten best and worst performing configurations of Experiment 2-2 on the Statlog Satellite dataset are tabulated in Table D.11 and D.12, respectively.

The prediction performance of the ten best configurations ranged from 89.40% to 89.15% Accuracy. Generally, the configurations were quite similar in learning_rate values for the different trees. For instance, the average of the first tree was 0.71, of which most configurations had a relatively close value, derived from the relatively low standard deviation.

The same can be said for the other trees. The 10 best configurations were thus similar, but clearly distinguishable.

The prediction performance of the ten worst configurations ranged from 85.35% to 86.25% Accuracy. Much like with the best configurations, the worst configurations were relatively similar. For instance, the average of the first tree was 0.10, of which most configurations had a relatively close value, derived from the relatively low standard deviation. The same can be said for the other trees, though Tree 3 and 4 seemed more variable, having standard deviations above 0.3. The 10 worst configurations were thus similar, but clearly distinguishable.

Comparing the best and worst configurations on the Statlog Satellite dataset, the learning_rate values were quite different, based especially on Tree 1, 2 and 5, where the average were considerably lower. These differences, as well as the similarities in characteristics between the configurations in each prediction performance group, indicate that the area of best performance in the learning_rate search landscape can be somewhat predicted and exploited. The fact that similar prediction performance was achieved with similar, but slightly different configurations, also indicate that the area of best performance is relatively lenient. Roughly the average plus/minus the standard deviation for each tree.

Prediction Performance	learning_rate				
Accuracy	Tree 1	Tree 2	Tree 3	Tree 4	Tree 5
89.40%	0.6297	0.5606	0.6656	0.5940	0.6475
89.35%	0.7016	0.4004	0.9085	0.9940	0.7722
89.35%	0.4031	0.9614	0.9478	0.2738	0.9559
89.30%	0.6405	0.6806	0.7525	0.9099	0.5454
89.25%	0.5870	0.6280	0.6077	0.8530	0.6518
89.25%	0.5151	0.5618	0.7061	0.8788	0.9832
89.20%	0.3713	0.5709	0.9270	0.1636	0.9943
89.20%	0.3865	0.8413	0.8306	0.8341	0.8293
89.20%	0.3871	0.7425	0.9352	0.5672	0.8332
89.15%	0.4109	0.5955	0.7579	0.6806	0.9731
Average:	0.7143	0.6543	0.8039	0.6749	0.8186
Standard Deviation:	0.1267	0.1601	0.1232	0.2781	0.1615

Table D.11: The 10 best flexible structure configurations for the Statlog Satellite dataset, based on cross validation scores from Experiment 2-2. The average learning_rate values and standard deviations for each tree are included.

Prediction Performance	learning_rate				
Accuracy	Tree 1	Tree 2	Tree 3	Tree 4	Tree 5
85.35%	0.0115	0.0651	0.0104	0.0264	0.2144
85.60%	0.0728	0.2073	0.0154	0.6814	0.0881
86.00%	0.0142	0.0064	0.9426	0.0661	0.0413
86.00%	0.0330	0.0167	0.9585	0.2144	0.0387
86.00%	0.0285	0.1065	0.1782	0.7172	0.0055
86.15%	0.0379	0.0583	0.2143	0.9103	0.3787
86.20%	0.5267	0.0007	0.0277	0.0589	0.4390
86.25%	0.1076	0.1970	0.9836	0.0884	0.0966
86.25%	0.1660	0.0355	0.1593	0.7996	0.0345
86.25%	0.0124	0.8591	0.4488	0.0349	0.0802
Average:	0.1011	0.1553	0.3939	0.3598	0.1417
Standard Deviation:	0.1576	0.2581	0.4122	0.3676	0.1525

Table D.12: The 10 worst flexible structure configurations for the Statlog Satellite dataset, based on cross validation scores from Experiment 2-2. The average learning_rate values and standard deviations for each tree are included.

D.7 Winequality-red

The ten best and worst performing configurations of Experiment 2-2 on the Winequality-red dataset are tabulated in Table D.13 and D.14, respectively.

The prediction performance of the ten best configurations were ranged from 68.75% to 67.81% Accuracy. Generally, the configurations were relatively similar in learning_rate values for the different trees. For instance, the average of the first tree was 0.69, of which most configurations had a relatively close value, derived from the low standard deviation. The same can be said for the other trees, though Tree 2 seemed a bit more variable, having a standard deviation above 0.3. The 10 best configurations were thus similar, but clearly distinguishable.

The prediction performance of the ten worst configurations ranged from 54.38% to 55.63% Accuracy. Much like with the best configurations, the worst configurations were relatively similar. For instance, the average of the first tree was 0.13, of which most configurations had a relatively close value, derived from the low standard deviation. The same can be said for the other trees, though Tree 2, 4 and 5 seemed more variable, having standard deviations above 0.3. The 10 worst configurations were thus relatively similar, but clearly distinguishable.

Comparing the best and worst configurations on the Winequality-red dataset, the learning_rate values were somewhat different based on the averages. These were generally slightly lower for the worst configurations. This indicates that the area of best performance in the learning_rate landscape might be somewhat difficult to predict and exploit with this dataset.

Prediction Performance	learning_rate				
Accuracy	Tree 1	Tree 2	Tree 3	Tree 4	Tree 5
68.75%	0.7345	0.2266	0.1696	0.5744	0.9483
68.44%	0.7188	0.9659	0.5016	0.2606	0.4672
68.44%	0.7257	0.9053	0.4448	0.8034	0.8930
68.13%	0.6122	0.7770	0.1698	0.6134	0.4078
68.13%	0.8391	0.4424	0.3853	0.7904	0.9422
68.13%	0.6381	0.0054	0.8078	0.2033	0.6488
68.13%	0.6277	0.8073	0.8595	0.2831	0.7305
67.81%	0.7586	0.8842	0.6014	0.8776	0.4579
67.81%	0.6511	0.8123	0.6908	0.2642	0.0782
67.81%	0.6376	0.7110	0.6877	0.8553	0.3538
Average:	0.6943	0.6537	0.5318	0.5526	0.5928
Standard Deviation:	0.0728	0.3212	0.2427	0.2757	0.2888

Table D.13: The 10 best flexible structure configurations for the Winequality-red dataset, based on cross validation scores from Experiment 2-2. The average learning_rate values and standard deviations for each tree are included.

Prediction Performance	learning_rate				
Accuracy	Tree 1	Tree 2	Tree 3	Tree 4	Tree 5
54.38%	0.0988	0.9985	0.1226	0.0432	0.3164
54.69%	0.0905	0.0395	0.6252	0.6906	0.5737
55.00%	0.1281	0.0177	0.8340	0.3300	0.6649
55.00%	0.1024	0.8798	0.3036	0.6985	0.1708
55.31%	0.1004	0.7277	0.1787	0.8585	0.0682
55.31%	0.0997	0.5753	0.3958	0.2960	0.2404
55.63%	0.1344	0.9880	0.2924	0.0097	0.1818
55.63%	0.3178	0.3007	0.3335	0.2010	0.9085
55.63%	0.1361	0.8955	0.2403	0.5106	0.9096
55.63%	0.1442	0.2210	0.0546	0.9118	0.3448
Average:	0.1352	0.5643	0.3381	0.4550	0.4379
Standard Deviation:	0.0670	0.3895	0.2349	0.3271	0.3073

Table D.14: The 10 worst flexible structure configurations for the Winequality-red dataset, based on cross validation scores from Experiment 2-2. The average learning_rate values and standard deviations for each tree are included.

Appendix E

Experiment 5 Per-Dataset Results

E.1 Concrete

The results of Experiment 2-3 for the Concrete dataset are tabulated in Table E.1. The best prediction performance obtained across all search processes was 3.8006 MAE, which was obtained with 2000 search iterations and quniform as the value selection method. Comparably, the best prediction performance with uniform was 3.8109 MAE, also obtained through 2000 iterations. Quniform obtained better prediction performance in all processes except the one with 1000 iterations. However, this this value, 3.8784, was worse that obtained in the 500 iteration process, being 3.84. Regarding the configurations obtained with uniform and quniform, these were quite similar in the 500 and 2000 iterations search processes, and relatively different in the 1000 iteration process.

Due to the fact that quniform obtained the best prediction performance, and better in two of the three search processes, it seemed like quniform was generally a more effective value selection method for the Concrete dataset. However, as much as 2000 iterations was still necessary to obtain the best prediction performance.

	500 Iterations		1000 Iterations		2000 Iterations	
	Uniform	Quniform	Uniform	Quniform	Uniform	Quniform
MAE	3.8946	3.8416	3.8281	3.8784	3.8109	3.8006
Tree 1 learning_rate	0.3227	0.32	0.3372	0.57	0.3210	0.35
Tree 2 learning_rate	0.5303	0.61	0.5833	0.66	0.5244	0.6
Tree 3 learning_rate	0.5281	0.56	0.4918	0.56	0.5120	0.52
Tree 4 learning_rate	0.7319	0.9	0.6342	0.89	0.7751	0.78
Tree 5 learning_rate	0.8567	0.84	0.9967	0.83	0.9491	0.94

Table E.1: The best MAE and flexible structure configuration, defined by learning_rate, for the uniform and quniform value selection methods, obtained through separate runs of 500, 1000 and 2000 iterations of Bayesian Optimization on the Concrete dataset.

E.2 Energy Prediction

The results of Experiment 2-3 for the Energy Predictin dataset are tabulated in Table E.2. The best prediction performance obtained across all search processes was 41.7235 MAE, which was obtained with 2000 search iterations and quniform as the value selection

method. Comparably, the best prediction performance with uniform was 41.7953 MAE, also obtained through 2000 iterations. Quniform obtained better prediction performance in all processes. Regarding the configurations obtained with uniform and quniform, these were generally quite similar.

Overall, it seemed like quniform was a more effective value selection method for the Energy Prediction dataset, but still needed as much as 2000 iterations to obtain the best found prediction performance.

	500 Iterations		1000 Iterations		2000 Iterations	
	Uniform	Quniform	Uniform	Quniform	Uniform	Quniform
MAE	41.9186	41.8425	41.8411	41.8388	41.7953	41.7235
Tree 1 learning_rate	0.2117	0.29	0.3017	0.3	0.3022	0.21
Tree 2 learning_rate	0.2974	0.24	0.2108	0.3	0.2054	0.27
Tree 3 learning_rate	0.2509	0.34	0.3243	0.27	0.3134	0.31
Tree 4 learning_rate	0.2473	0.27	0.3469	0.18	0.2784	0.27
Tree 5 learning_rate	0.3007	0.24	0.2101	0.23	0.2532	0.29

Table E.2: The best MAE and flexible structure configuration, defined by learning_rate, for the uniform and quniform value selection methods, obtained through separate runs of 500, 1000 and 2000 iterations of Bayesian Optimization on the Energy Prediction dataset.

E.3 Housing

The results of Experiment 2-3 for the Housing dataset are tabulated in Table E.3. The best prediction performance obtained across all search processes was 2.3175 MAE, which was obtained with 2000 search iterations and quniform as the value selection method. Comparably, the best prediction performance with uniform was 2.3210 MAE, also obtained through 2000 iterations. Quniform obtained better prediction performance in all processes except the one with 500 iterations. Regarding the configurations obtained with uniform and quniform, these were generally quite similar in the 1000 and 2000 iteration search processes, and relatively different in the 500 iteration process.

Due to the fact that quniform obtained the best prediction performance, and better in two of the three search processes, it seemed like quniform was generally a more effective value selection method for the Housing dataset. However, as much as 2000 iterations was still necessary to obtain the best prediction performance.

	500 Iterations		1000 Iterations		2000 Iterations	
	Uniform	Quniform	Uniform	Quniform	Uniform	Quniform
MAE	2.3813	2.4047	2.3596	2.3397	2.3210	2.3175
Tree 1 learning_rate	0.3161	0.21	0.3253	0.2	0.1926	0.2
Tree 2 learning_rate	0.4407	0.58	0.4542	0.55	0.5252	0.48
Tree 3 learning_rate	0.5836	0.45	0.5256	0.56	0.5141	0.53
Tree 4 learning_rate	0.8792	0.96	0.9619	0.77	0.4792	0.44
Tree 5 learning_rate	0.8009	0.56	0.7894	0.85	0.9836	0.92

Table E.3: The best MAE and flexible structure configuration, defined by learning_rate, for the uniform and quniform value selection methods, obtained through separate runs of 500, 1000 and 2000 iterations of Bayesian Optimization on the Housing dataset.

E.4 Seoul Bike Sharing

The results of Experiment 2-3 for the Seoul Bike Sharing dataset are tabulated in Table E.4. The best prediction performance obtained across all search processes was 159.64 MAE, which was obtained with 2000 search iterations and quniform as the value selection method. Comparably, the best prediction performance with uniform was 159.76 MAE, also obtained through 2000 iterations. Quniform obtained better prediction performance in all processes on this dataset. Regarding the configurations obtained with uniform and quniform, these were generally quite similar.

Overall, it seemed like quniform was a more effective value selection method for the Seoul Bike Sharing dataset, but still needed as much as 2000 iterations to obtain the best found prediction performance.

	500 Iterations		1000 Iterations		2000 Iterations	
	Uniform	Quniform	Uniform	Quniform	Uniform	Quniform
MAE	161.09	160.42	160.44	160.23	159.76	159.64
Tree 1 learning_rate	0.5152	0.49	0.3617	0.5	0.5048	0.5
Tree 2 learning_rate	0.6162	0.63	0.6838	0.61	0.5888	0.61
Tree 3 learning_rate	0.6825	0.62	0.6303	0.63	0.6174	0.61
Tree 4 learning_rate	0.8011	0.87	0.6594	0.73	0.7708	0.6
Tree 5 learning_rate	0.8822	0.83	0.8798	0.82	0.8807	0.87

Table E.4: The best MAE and flexible structure configuration, defined by learning_rate, for the uniform and quniform value selection methods, obtained through separate runs of 500, 1000 and 2000 iterations of Bayesian Optimization on the Seoul Bike Sharing dataset.

E.5 Car Evaluation

The results of Experiment 2-3 for the Car Evaluation dataset are tabulated in Table E.5. The best prediction performance obtained across all search processes was 0.015625 Error, which was obtained with 2000 search iterations and uniform as the value selection method. However this was only a slightly better value than the best obtained with quniform, being 0.015628 Error, which was obtained in both the 1000 and 2000 iteration processes. Quniform obtained better prediction performance in all other processes beyond the 2000 iteration one,

which had and arguably insignificant difference. Regarding the configurations obtained with uniform and quniform, these were generally quite similar across all search processes.

Even though the best prediction performance was obtained with uniform, the difference between this Error and the best obtained with quniform was arguably insignificant. The best prediction performance with quniform was additionally obtained with 1000 less iterations. Therefore, quniform was clearly competitive with uniform, and arguably better, as a value selection method for the Car Evaluation dataset.

	500 Iterations		1000 Iterations		2000 Iterations	
	Uniform	Quniform	Uniform	Quniform	Uniform	Quniform
Error	0.018808	0.018230	0.016786	0.015628	0.015625	0.015628
Tree 1 learning_rate	0.8435	0.91	0.9065	0.91	0.8056	0.91
Tree 2 learning_rate	0.9377	0.98	0.9774	0.98	0.9078	0.96
Tree 3 learning_rate	0.8295	0.91	0.9819	1.0	0.9839	0.98
Tree 4 learning_rate	0.9166	0.87	0.9805	0.95	0.9846	0.95
Tree 5 learning_rate	0.8243	0.77	0.9459	0.84	0.9995	0.94

Table E.5: The best Error and flexible structure configuration, defined by learning_rate, for the uniform and quniform value selection methods, obtained through separate runs of 500, 1000 and 2000 iterations of Bayesian Optimization on the Car Evaluation dataset.

E.6 Statlog Satellite

The results of Experiment 2-3 for the Statlog Satellite datset are tabulated in Table E.6. The best prediction performance obtained across all search processes was 0.099456 Error, which was obtained with 500 search iterations and uniform as the value selection method. Comparably, the best prediction performance with quniform was 0.099844, obtained trough 1000 iterations. Quniform obtained worse prediction performance in all processes. A noteworthy observation was that prediction performance did not seem get better with more search iterations, but seemed quite random. Regarding the configurations obtained with uniform and quniform, these were generally quite similar in the 1000 iteration search process, but relatively different in the 500 and 2000 iterations processes.

Generally it seemed like quniform was a less effective value selection method than uniform for the Statlog Satellite dataset.

	500 Iterations		1000 Iterations		2000 Iterations	
	Uniform	Quniform	Uniform	Quniform	Uniform	Quniform
Error	0.099456	0.100777	0.099689	0.099844	0.099689	0.100466
Tree 1 learning_rate	0.3528	0.3	0.3815	0.38	0.3537	0.27
Tree 2 learning_rate	0.5744	0.8	0.7722	0.79	0.5342	0.78
Tree 3 learning_rate	0.9461	0.76	0.9589	0.94	0.8221	0.97
Tree 4 learning_rate	0.8592	0.83	0.9228	1.0	0.9077	0.89
Tree 5 learning_rate	0.8051	0.92	0.8782	1.0	0.7003	0.83

Table E.6: The best Error and flexible structure configuration, defined by learning_rate, for the uniform and quniform value selection methods, obtained through separate runs of 500, 1000 and 2000 iterations of Bayesian Optimization on the Statlog Satellite dataset.

E.7 Winequality-red

The results of Experiment 2-3 for the Winequality-red dataset are tabulated in Table E.7. The best prediction performance obtained across all search processes was 0.343045 Error, which was obtained with 2000 search iterations and quniform as the value selection method. Comparably, the best prediction performance with uniform was 0.343666, also obtained through 2000 iterations. Quniform obtained better prediction performance in all processes except the one with 1000 iterations. regarding the configurations obtained with uniform and quniform, these were generally quite similar in the 1000 and 2000 iteration search processes, and quite different in the 500 iteration process.

Due to the fact that quniform obtained the best prediction performance, and better in two of the three search processes, it seemed like quniform was generally a more effective value selection method for the Winequality-red dataset. However, as much as 2000 iterations was still necessary to obtain the best prediction performance.

	500 Iterations		1000 Iterations		2000 Iterations	
	Uniform	Quniform	Uniform	Quniform	Uniform	Quniform
Error	0.346778	0.343972	0.344907	0.347097	0.343666	0.343045
Tree 1 learning_rate	0.3624	0.52	0.4231	0.42	0.4633	0.46
Tree 2 learning_rate	0.8996	0.34	0.8410	0.73	0.5977	0.57
Tree 3 learning_rate	0.8225	0.76	0.9832	0.98	0.9209	0.88
Tree 4 learning_rate	0.7145	0.96	0.9369	0.9	0.9997	0.96
Tree 5 learning_rate	0.7881	1.0	0.9553	0.56	0.9373	1.0

Table E.7: The best Error and flexible structure configuration, defined by learning_rate, for the uniform and quniform value selection methods, obtained through separate runs of 500, 1000 and 2000 iterations of Bayesian Optimization on the Winequality-red dataset.