https://doi.org/10.31449/inf.v47i9.3844 Informatica **47** (2023) 17–34 **17**

A Consolidated Tree Structure Combining Multiple Regression Trees with Varying Depths, Resulting in an Efficient Ensemble Model

Elmira Ashoor Mahani*, Koorush Ziarati
Comp. Sci. & Eng. & I.T. Dept., Shiraz University, Mollasadra Street, Shiraz, Iran
E-mail: elmira.ashoor@cse.shirazu.ac.ir, ziarati@shirazu.ac.ir
*Corresponding Author

Keywords regression tree, stopping condition, ensemble, efficient

Received: November 28, 2021

Regression is a commonly used technique to predict a continuous target value based on a set of input features. Decision trees are hierarchical models that offer high interpretability, fast and precise reasoning, and are also used for regression tasks. However, determining the optimal stopping conditions for decision trees is a complex problem that has attracted significant research interest. Ensemble based modeling is an effective approach for adjusting hyper-parameters, where base models with varying parameter values are combined instead of searching for the best value. Random forests are a classic example of an ensemble model that combines decision trees generated from different perspectives. This paper proposes a novel approach that generates base trees using the same tree-generation procedure, but with different stopping conditions. Unlike random forests, this model can be efficiently integrated into a single tree structure. Additionally, the paper proposes some aggregation methods based on weighting the base models. Experimental results on standard datasets demonstrate that the proposed method outperforms well-known stopping conditions.

Povzetek: Razvita je nova metoda kombiniranja regresijskih dreves, ki dosega boljše rezultate v primerjavi z znanimi metodami v regresijskih nalogah.

1 Introduction

Decision trees are commonly used tools for pattern recognition, especially in supervised tasks like classification and regression. Their hierarchical structure allows for simple, interpretable, fast, and accurate decision-making. Decision trees have been extensively researched in the literature, with many applications focusing on classification tasks such as intrusion detection, privacy preservation, power systems, bank marketing, health care and disease diagnosis, and agriculture [1-7]. Additionally, researchers have investigated regression models based on decision trees, which are typically referred to as regression trees, in fields like psychology, education, urban planning, environmental management, genetics, communication, and economics [8-15].

In Table 1, the regression methods are compared. The popularity of decision trees can be attributed to their ability to provide a simple and interpretable representation of the data, which is particularly important in fields where understanding and analyzing the model's output is crucial. Furthermore, decision trees are fast and accurate, making them attractive for real-time applications. However, their performance can be affected by the quality of the data used to train the model, and they may not always be the best choice for complex datasets. Despite these limitations, decision trees remain a valuable tool for pattern recognition and continue to be an active area of research.

1.1 Decision tree

A decision tree is a flowchart-like structure with a tree hierarchy, where each internal node includes a test on a feature, each branch represents the output of the test, and each leaf node represents a consequence. To make a decision on a given instance using a decision tree, the instance must traverse a complete path from the root to a leaf in the tree. At the leaf, a value or a model is provided to predict the target value of the instance, such as a specific classifier or regression model. Decision trees can easily be converted into a set of rules, where each path from the root to a leaf is a rule, making them simple and completely interpretable.

Compared to other rule-based systems, decision trees are efficient in decision-making due to their hierarchical structure, and not all rules need to be checked. Additionally, decision trees can simultaneously divide the feature space into small or large regions, providing specific and general rules, respectively. This ability makes decision trees one of the most accurate prediction models. In summary, decision trees are simple, interpretable, fast, and accurate models that can efficiently divide the feature space and convert the tree structure into a set of rules.

The time complexity of decision tree induction has been shown to increase exponentially with the height of the tree. To overcome this limitation, heuristic methods have been proposed in the literature to produce shallow, small, and/or accurate trees [13,14]. These methods aim to optimize a predefined objective function on the training set by partitioning the feature space into two or more sub-regions based on a specific feature. The resulting tree includes internal (decision) nodes and leaves, where each node is associated with a region that may be partitioned based on a feature or labeled as a leaf node.

In a classic decision tree, each sub-region, considering associated instances, is recursively partitioned into smaller ones until a stopping condition is satisfied. The partitioning process forms a tree that provides a hierarchical representation of the decision-making process. At each internal node, a test is performed on a feature (called pivot feature) to partition the feature space into sub-regions. The objective is to maximize the homogeneity of the instances within each sub-region with respect to the target variable. The partitioning process continues until a stopping condition is met, typically when a sub-region contains a small number of instances or when further partitioning does not improve the accuracy of the model. In regression models, the best feature is the one that can reduce the error metric of regression (e.g., mean square error) on the training instances more than others.

In a regression tree, if a region cannot be further partitioned and some stopping conditions are met, the partitioning process is stopped. The instances associated with that region are then used for reasoning. A single value may be assigned as the prediction value for all instances in the leaf node, or a regression model trained on the associated instances may be used for future prediction. The assigned value or model represents the prediction of the decision tree for instances that belong to that region in the feature space.

After the tree construction phase, the resulting tree may have many leaves, particularly in the presence of noisy data and outliers. This can lead to overfitting, where the tree is too complex and fits the training data too closely, even capturing undesired noises. To address this issue and remove the least reliable branches, statistical measures are typically used to prune the tree.

Pruning the decision tree results in a smaller, less complex, and easier-to-understand tree, which is usually faster and more accurate in classifying test data. Pruning can be performed either pre-pruning or post-pruning, depending on the stopping conditions used in the tree construction phase. Pre-pruning involves setting stopping conditions during tree construction to avoid overfitting, while post-pruning involves removing branches after construction based on statistical measures. Proper stopping conditions can lead to an efficient pre-pruning approach that is more accurate and faster than post-pruning [15].

In the testing phase, a query instance is fed into the decision tree starting from the root node and following a path through the tree until a leaf node is reached. At each node, the value of the query instance for the associated pivot feature is used to determine which child node to traverse to in the next layer. This process continues until the query instance reaches a leaf node. In the leaf node, the assigned value or model is used to predict the corresponding target value of the query instance.

1.2 Random forest

Although decision trees have a simple structure and provide acceptable performance, finding the optimal tree that optimizes the desired objective function can be a challenging task. The construction process is typically done using a greedy method, which can result in a suboptimal tree. As a result, small changes in the parameters, distribution of training instances, or objective function can significantly alter the final tree. To address this issue, ensemble methods have been proposed to generate multiple trees from different perspectives to improve the stability and accuracy of the model [16].

Ensemble methods use multiple base models that can differ in type, model, hyperparameters, construction method, and instance set used for training. In the case of using a set of decision trees as the base models, the resulting ensemble model is called a Random Forest (RF). There are various types of RFs that use different approaches to generate the individual decision trees, such as Adaboost, boost strap or bagging [17-20].

Decision trees themselves have the ability to divide the feature space into subregions and their associated instances, which in turn allows a dedicated subtree to be assigned to each region to recursively classify the associated training instances. Therefore, decision trees employ a specific instance selection to some extent. In RF, the base trees typically differ in the random features chosen in each step to partition the feature space. This is known as feature bagging [21]. Additionally, the random selection of splitting points in each node is another characteristic of classic RF algorithm [22]. These randomization techniques help to reduce overfitting and increase the diversity of the base models, which in turn improves the accuracy and stability of the ensemble model.

In ensemble methods such as Random Forests (RFs), a majority vote (in classification) or averaging (in regression) is used to combine the predictions of all the trees on a query instance to make the final prediction. This requires storing all the trees. Also, the output of all the trees should be determined for final prediction as a time-consuming task during the testing phase. Moreover, a large number of decision trees can degrade the interpretability of the model. Some rules may be generated whose antecedents are satisfied, but their consequences differ significantly from the final outcome of the ensemble model due to the aggregation of the consequences of all activated rules. Despite these limitations, RFs have been shown to significantly improve the performance of decision trees, and they remain popular in many applications. In this paper, trees ensembled with different stopping conditions is considered, which can lead to a more efficient and interpretable model.

1.3 Stopping conditions

During the construction of a decision tree, the expansion of a node may be stopped for various reasons, such as decreasing memory complexity or preventing overfitting on the training data. To prevent overfitting and control the complexity of the decision tree, a set of hyperparameters is used to define and evaluate the stopping conditions. The stopping condition is checked at each node to decide whether to expand the node further or stop partitioning. Some stopping conditions' hyperparameters are reported in Table 2. The ones investigated in this paper are:

- Maximum Depth of the tree (MxD): This parameter limits the maximum allowed depth of the decision tree. If the depth of a node exceeds this value, the node expansion is stopped.

- Minimum number of Instances per leaf (MnI): This parameter sets the minimum number of training instances required to split a node. If a node has fewer instances than this threshold, the node expansion is stopped.

- Maximum number of Leaves (MxL): This parameter limits the maximum allowed number of leaf nodes in the decision tree. If the number of leaves exceeds this value, the node expansion is stopped.

- Maximum Error per leaf (MxE): This parameter sets the maximum error can be handled in a leaf. If the error exceeds this threshold, the leaf should be expanded.

-Minimum (Relative) Promotion (MnP/MnRP): Each expansion should be along with a minimum decrease in the total error to be done. The value of decrease in error may be relatively computed respect to the current error.

These hyperparameters can be tuned to optimize the performance of the decision tree and prevent overfitting. The merits and drawbacks of them are discussed later in this paper. The optimal set of hyperparameters depends on the characteristics of the training data and the desired performance of the model. On the other hand, adjusting the hyperparameters of a decision tree is a time-consuming and challenging task, and there is no rule of thumb to determine the optimal values for any given dataset. Cross-validation on the training data is a common technique for hyperparameter tuning in machine learning models. However, cross-validation can be expensive, and separating a set of validation data from the training set can result in variations in the constructed decision trees such that the best hyperparameters may not be suitable with sufficient confidence. In addition, cross-validation is a task of evaluating a value for a hyper parameter. A search strategy is required to extract candidates and find the optimal solution. The strategies will be compared later.

Moreover, different regions of the feature space may require different attention to extract the decision boundaries. However, the hyperparameters of the stopping conditions affect all the nodes in a single tree. Therefore, in some cases where more specificity is required, the expansion may be stopped, while in another branch of the tree, generalization may be sacrificed for extra partitioning. Additionally, the importance of the stopping conditions may vary from case to case.

1.4 Motivations and innovations

In this paper, a novel approach is proposed to ensemble learning using decision trees with different hyperparameters, specifically focusing on the depth of the trees. This approach simplifies the process of designing these hyperparameters by generating many trees with different stopping conditions, which are then aggregated to produce the final result. In this approach, the hyperparameters are integrated into a single, simple solution by constructing trees with different depths in each branch. Finally, the resulting trees are merged into a single tree, called as the Mother Tree. This approach preserves both the efficiency and interpretability of the model, since it occupies memory only as large as a single tree and allows the results of all the virtual trees to be computed for a query instance by a single pass on the associated path from the root to a leaf of the Mother Tree. The proposed approach offers a simple and effective solution for stopping condition adjustment based on ensemble learning that overcomes the limitations and challenges of traditional decision trees and random forests, and provides an interpretable model.

Weighted model aggregation has been shown to improve the performance of ensemble methods, but it can also be computationally expensive, especially when dealing with a large number of trees. Heuristic approaches, such as Adaboost are commonly used to address this issue [18]. In this paper, some simple weighting approaches are proposed, which can be applied to the Mother Tree structure. The proposed approaches assign weights to the virtual trees in voting phase, each one with its own characteristics.

The remainder of this paper is organized as follows. In Section 2, a review of the related work is given on ensemble learning and model aggregation. In Section 3, the proposed model is presented. In Section 4, the experimental results are reported followed by discussions. Finally, the paper is concluded in Section 5

2 Related work

A regression tree is a variant of the classification decision tree that predicts a real value instead of a class label. In a regression tree, a constant value or a regression model is created at each leaf node after constructing the tree. The node error is usually calculated as the mean square of all differences between the desired and predicted values for all validation instances in that node. The first known regression tree is the AID method [23] . AID declares the mean of target values of all instances reached at each leaf node as its regression value. Hence the mean-square error is equal to the variance of associated target values as the impurity measure of each node. This method generates a piecewise constant model that has good interpretability but may have lower accuracy than models with more smoothness.

M5' is a regression tree that uses an efficient strategy to create a piecewise linear model [24]. The first output of this strategy is a piecewise constant tree, and then a proper linear regression is found and replaced constant value for

each leaf. Because the tree structure is piecewise constant, the M5' final tree is larger than other piecewise linear trees. Fidalgo-Merino et al. introduced an incremental algorithm to generate a tree [25]. However, most recent research has applied Classification and Regression Trees (CART) as the base regression model more than other models in the literature [26].

Regression trees are typically constructed using heuristics, which do not guarantee the optimality of the output. Kordos et al. introduced an evolutionary approach in which a set of regression trees evolves to achieve the best tree [27]. Two important evolutionary operators, mutation and crossover, were implemented by substituting selected attributes along branches and exchanging subtrees between different trees. However, evolutionary search methods can be time-consuming, and defining a fitness function that can accurately describe the optimal tree may not be straightforward. With such ambiguity, finding the tree with the optimum objective function may not be necessary.

Due to the uncertainties surrounding the proper objective function, limitations, and structure of regression trees, researchers have turned to ensemble different views of tree construction. RSSCARD is a method proposed in to predict spatial landslides [28]. They hybridized Random Subspaces (RSS) and CART to achieve their goal. Choubin and et al. ensemble, for the first time, CART and Multivariate Discriminant Analysis (MDA) to analyze flood susceptibility and obtained acceptable results [29]. Random Forests (RF) are also an ensemble of decision trees that combine the merits of different structures or parameters. Decision trees are faster to train than RFs, which is why they are still desirable. However, the accuracy of Random Forests (RF) is greater [30].

Parameter adjustment in tree construction is also a challenge in this research field. One of the most important sets of parameters are the ones associated with the stopping conditions, such as maximum depth, maximum leaves, and maximum error. Bing Zhang et al. used CART to predict blood pressure, and they used cross-validation to avoid overfitting, choose the best parameters, and develop a better general model [31]. Ensemble methods, including RFs, can also be used to aggregate trees with different parameters. In this case, time-consuming cross-validation is not used, and the randomness of the model injected from cross-validation is removed. However, to our knowledge, this ensemble approach has not yet been used in the literature as a solution for the problem of stopping conditions, especially for regression models. In this paper, not only is this ensemble approach presented, but also it has been efficiently integrated into the structure of a single tree called the Mother Tree. Finally, instead of generating many trees, a reasoning strategy has been proposed based on the generated Mother Tree with various aggregation methods.

3 Proposed method

The main contributions of this paper, as mentioned previously, are as follows:

1. All stopping conditions and parameter combinations are integrated into one parameter, Maximum Depth of tree (MxD), as the only difference among the candidate trees. It has been proven that MxD is a sufficient parameter as the stopping condition, here.
2. All constructed trees are merged into a single tree with the time-complexity of generating just one tree. This approach saves computational resources and improves the interpretability of the model.
3. Some weighted average methods are proposed on this integrated tree to ensemble all candidate trees. These methods provide more robust and accurate prediction by combining the outputs of multiple trees. Each aggregation method has its own attributes and behavior on datasets.

Each one of these contributions is addressed in detail in the following subsections.

3.1 Maximum depth of tree

Stopping conditions' (SC) parameters do not affect the selection of the pivot feature of each node or the best set of splitting points in the nodes. However, these parameters can determine whether a node should be expanded or not. As a result, some leaves may be generated based on SCs that are internal nodes in some other trees (due to later stopping) and may not be generated in other trees (due to early stopping). The training set reached at an internal node or leaf is deterministic and unique in all possible trees that have that node. If the parameter of the stopping condition is changed, it can alter the tree's depth of some branches. The mostly expanded possible tree is called, here, the Mother Tree (MT). However, the MT may also follow some hard hyper restrictions defined by experts which prevent more expansion, such as the Maximum Depth of Tree. For example, consider an MT constructed as shown in Figure 1, where the Sum Square Error (SSE) of the target values of belonging instances in each node is also reported in the nodes.

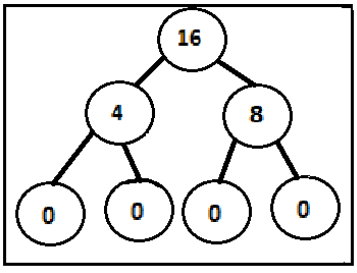

Figure 1: An example of Mother Tree expanded to reach zero sum square error in leaves.

If Maximum Error (MxE) in leaves is considered as the stopping condition, each node with SSE greater than ME will be expanded, whereas nodes with SSE less than or equal to MxE will become leaves. Decreasing this upper-bound may lead to a deeper tree by expanding some nodes until reaching the MT shown in Figure 1. Other intermediate possible trees associated with different values of MxE are depicted in Figure 2. In this example, MT is completely generated for MxE < 4.

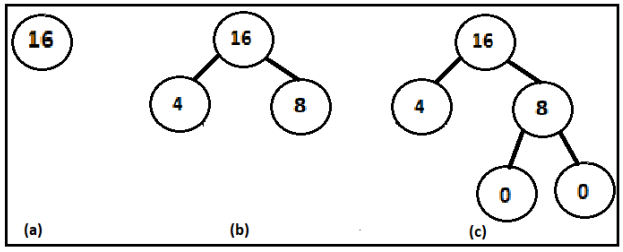

Figure 2: Intermediate tree structures depicted in Fig. 1: (a) MxE ≥ 16, (b) MxE ∈ [8,16), (c) MxE ∈ [4,8).

It is evident that only full rooted subtrees (tree structures) can be generated because when an internal node is expanded, all of its children are generated. Cross-validation is a common method for adjusting hyperparameters to the best possible value in a candidate set. However, there is often no guarantee that two different candidate values will generate different tree structures. Additionally, a value that generates a specific tree structure may not be included in the candidate set. For example, cross-validation on the candidate values of MxE {18, 13, 8, 3} can only generate two out of the three tree structures in Figure 2 and the Mother Tree in Figure 1. In other words, MxE = 13 or 8 generates similar structures, and the one shown in Figure 2-c is missed. However, a complete set of non-redundant candidates can be generated in this case by sorting all positive SSEs in the nodes of the MT.

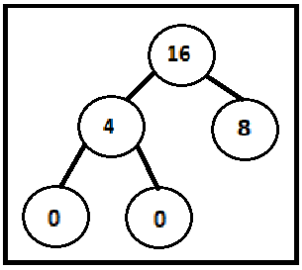

Figure 3. A tree structure of MT in Fig.1 which is not feasible with just MxE as the stopping condition.

The main problem, however, remains. Given a predefined parametric stopping condition, some node expansions are inevitable to achieve other specific expansions. For example, by decreasing MxE of the tree, it cannot be controlled to have expansion in specific nodes only. Hence, generating all tree structures is not guaranteed. For instance, the tree structure depicted in Figure 3 cannot be generated with this stopping condition because if a node with SSE = 4 is expanded, the node with SSE = 8 should also be expanded. However, these nodes cover disjoint regions of the feature space with their unique characteristics and requirements.

Hybrid stopping conditions may generate more tree structures. There are many parameters associated with stopping conditions, as mentioned in the introduction, and each one can have numerous values if the parameter is discrete (e.g., MxD, MnI, and MxL) or a range of values if the parameter is continuous (e.g., MxE and MnP). Selecting an appropriate set of parameters and candidate value set for each one to construct all possible structures is likely impractical. Thus, conducting a complete search on tree structures is a challenging task, and the optimal structure may be missed.

Any parameter of the stopping condition only affects the lengths of the paths from the root to the leaves. In other words, these parameters are not used in reasoning and only determine the expandability of each node during the training phase. Once the nodes that should not be expanded (leaves) are determined, the tree structure can be uniquely constructed, and the target value for each query instance can be predicted. The tree structures are finite, countable, and much less than all combinations of the parameter values. As the first contribution of this paper, all tree structures are investigated regardless of any stopping condition to address the above concerns. However, the number of tree structures also exponentially increases with respect to the Maximum Depth of the tree.

Assume that the Mother Tree is a full binary tree with depth D at all leaves, called here a D-tree. For D-trees with depth D = 1 and 2, there are two and five tree structures, respectively. The five tree structures of the MT in Figure 1 are itself and four other structures presented in Figures 2 and 3. The number of tree structures can be very large for deeper trees. For example, a 4-tree, 5-tree, and 6-tree have 677, 458,330, and approximately 210 × 10^9structures, respectively. Such a large number of structures can make constructing all possible structures computationally infeasible. To address this problem, two techniques have been employed. Firstly, it is proved that a small set of tree structures defined with just one stopping condition, the Maximum Depth of tree, is sufficient to include an optimal structure for each query instance. Secondly, these structures are integrated to achieve a single structure without losing performance. These techniques significantly reduce the number of structures and make it practical to construct an optimal tree structure.

3.1.1 Minimum covering set

Determining the best tree structure with minimum total error on a set of validation data can be challenging. Regardless of the definition of the error function, the optimal tree structure is not necessarily the best for all query instances. Finding the tree structure with the minimum error for each query instance q in the validation set can help to use the strengths and overcome the weaknesses of different tree structures. This information can be used to select the best tree structure for a particular query instance or to develop an ensemble of tree structures that performs well on a diverse set of query instances.

Since a query instance q deterministically moves from the root of the MT down to reach the leaf, its path in its best structure is also determined to reach the associated leaf L. In other words, one tree structure in the set of d-tree structures ($d = 0, \dots, D$), includes L in associated path of q as the leaf and performs as well as the best structure for q.

Therefore, a set of $(D + 1)$ d-structures $(d = 0,1, \dots, D)$ can work as well as all exponential number of

tree structures. This approach significantly reduces the number of tree structures that need to be constructed and evaluated, making it computationally feasible to find an optimal tree structure.

3.2 Merging the trees

Assuming a Mother Tree (MT) is constructed and given, as shown in Figure 1. For each query instance q, the corresponding leaf in its optimal tree is denoted by $N^*(q)$. Tracking the associated path from the root towards the corresponding deepest node (leaf) in MT certainly passes $N^*(q)$. If the prediction value or model for all internal nodes of MT in this path is also computed, as well as the leaves, the query can compute its predicted target value by all d-structures. Therefore, just one tree structure (Mother tree) is sufficient to be constructed, whereas all nodes (internal and leaves) are assigned a prediction value or model in the training phase. The only problem is determining the optimal depth for a given query instance.

In this case, two different paths, separated from a disjoint node, can have their own stopping conditions, which are dynamically discovered in the test phase. Even two query instances that belong to a common path may have their own stopping condition on the tree. As a postponed decision, all prediction values of a query instance in all depths of MT can be gathered, and one of these candidate values can be assigned as the final prediction (e.g., the most frequent or the median). This approach can significantly reduce the computational cost of constructing multiple tree structures to generating just one tree but with a decision value or model in all the nodes.

In the training phase, it is assumed that the stopping condition may be held in each of the nodes (even in the root). Hence, each node is temporarily considered as a leaf, and a label or model is assigned to it. In the test phase, each given query instance q is fed to the MT to find its complete path. This path is a sequence of $D+1$ decision nodes $n_0(q), n_1(q), \ldots, n_D(q)$. Based on that, $D+1$ different predictions are computed for q representing the values or output of the models assigned to decision nodes. Now, it is time to reason for the final value (adaptively select the leaf or stopping condition). For example, the most frequent or median value in the predicted values can be assigned. This adaptive approach can improve the performance of the MT and make it more robust to different query instances.

3.3 Ensemble of tree structures

As mentioned earlier, the best tree structure for a query instance q exists; however, it may not be identified during the test phase. Even, its prediction for the target value may not be sufficiently accurate. In this case, none of the candidate values in the path are sufficiently accurate. To address this issue, this paper, similar to random forests, uses the average of the outputs of the regression trees for reasoning during the testing phase.

As the third contribution, this paper employs an ensemble of d-structures for reasoning, instead of selecting just one tree structure. The ensemble of $(D+1)$ d-structures is used for decision here. During the testing phase, each query instance q is fed to all $(D+1)$ d-structures in the ensemble, and $D+1$ different predictions are computed for q, as explained earlier. Then, the average of these predictions is returned as the final prediction for q. Also, a weighted approach can also be used as shown in (1).

$$y(q) = \frac{\sum_d (w_d * y_d(q))}{\sum_d w_d} \quad (1)$$

where $y_d(q)$ is the predicted value for query instance q by the d^{th} d-structure, w_d is the weight assigned to the d^{th} d-structure, and $\sum_d (w_d)$ is the sum of weights over all d-structures to normalize the average. Each node n_d contributes to the total reasoning as much as its positive weight w_d. If all d-structures have an equal effect on the total prediction, w_d is set to one.

3.4 Weighting the nodes

In this paper, some proper greedy weighting methods are sought for the nodes of a path to determine the final decision. As a classic approach, a statistical representative of the outputs of the path can be considered as the final approach as follows:

1. If it is desired to minimized the maximum difference between each output and the representative, the **Middle** point as the mean of minimum and maximum values is returned as the final decision. This output is robust against small changes in all outputs except of the minimum and maximum. However, it is highly sensitive to outlier outputs.
2. If it is desired to minimize sum of absolute differences between the outputs and the representative, final result is the **Median** of the outputs in the path. In contrary with Middle, Median is more robust against outliers and increasing maximum or decreasing minimum does not change its value.
3. Minimizing sum square of differences leads to return the **Mean** of outputs. It is sensitive to all values but as much as one vote between all other votes.

All of above approaches can be considered as a version of weighted average as shown in equations (2)-(4).

$$Middle = \frac{1}{2} v_0 + \frac{1}{2} v_D \quad (2)$$

$$Median = \frac{1}{2} v_{\lfloor \frac{D}{2} \rfloor} + \frac{1}{2} v_{\lfloor \frac{D}{2} \rfloor} \quad (3)$$

$$Mean = \frac{1}{D+1} \sum_{d=0}^{\mathcal{D}} v_d \quad (4)$$

where values v_d $(d = 0,1, \ldots, \mathcal{D})$ are sorted outputs of the path $y_d(q)$. However, the weights can be assigned

with other strategies respect to different goals. Two simple opposite strategies are given here.

Increasing weighting: In this strategy, it is assumed that deeper nodes (more specific rules) are more trustable due to their high resolution of looking training data. In this strategy, deeper leaves have greater weights. With a constant growth rate α, the weights $\alpha, 2\alpha, 3\alpha, \dots, (\mathcal{D}+1)\alpha$ are assigned to $w_0, w_1, w_2, \dots, w_{\mathcal{D}}$, respectively.

Decreasing weighting: From this point of view, the deepest node (the leaf) is not necessarily the best for reasoning, especially if the Mother Tree is permitted to be expanded as much as possible (which may result in overfitting to the training data). In addition, the training instances that belong to a leaf not only affect the output of the leaf but also play a role in determining the output of all internal nodes in the path. Therefore, the accumulated effect of training instances that belong to the leaf is more significant than that of others in determining the final result. This is why; to balance the effects, the weight of deeper nodes should be smaller. In this paper, reverse of previous approach is used and $\alpha, 2\alpha, \dots, (\mathcal{D}+1)\alpha$ are assigned to $w_{\mathcal{D}}, w_{\mathcal{D}-1}, \dots, w_0$, respectively.

In increasing weighting, each weight w_d should be normalized by equation (5).

$$w_d^N = \frac{2w_d}{\alpha(\mathcal{D}+1)(\mathcal{D}+2)} = \frac{2(d+1)\alpha}{\alpha(\mathcal{D}+1)(\mathcal{D}+2)} = \frac{2(d+1)}{(\mathcal{D}+1)(\mathcal{D}+2)} \quad (5)$$

where w_d^N is normalized value of w_d. Hence, α has no effect on the final result and the weights can similarly considered 1, 2, …,$\mathcal{D}+1$.

3.5 Discussions

In this subsection, some properties of the proposed model are discussed from different viewpoints. The proposed ensemble approach has several desirable properties that make it a promising method for solving classification and regression tasks.

The proposed ensemble approach can be considered as an ensemble model of $(D+1)$ d-structures with weights $w_0, w_1, \dots, w_D$, respectively. Each d-structure corresponds to a tree with a constant depth d in all branches.

In addition, the proposed model is flexible and can be adapted to different types of datasets and tasks. There is no parameter introduced by this paper for implementing the proposed ensemble model. The only parameters are related to the construction of the MT, such as the number of branches of an internal node for continuous pivot features or the maximum depth the MT can be expanded to. These parameters can be adjusted based on the specific characteristics of the dataset and the computational resources available.

3.5.1 Extra expansion

One of the challenges in using the proposed ensemble approach is determining the value of the maximum depth D of the MT. The value of D can be chosen based on the characteristics of the dataset and the computational resources available.

It is worth noting that the maximum depth D is equivalent to the Maximum Depth of the tree (MxD) used in a regression tree with a single stopping condition. In both cases, the tree is expanded up to a certain depth. However, the main difference between a regression tree with a single stopping condition based on MxD and the proposed method is in reasoning. In the former, the target value is predicted by just using the value or model assigned to the leaf node corresponding to the query instance. However, in the proposed method, a weighted average of predictions in the leaf node and its ancestors is used as the final result. This weighted average takes into account the contribution of each node to the final prediction. In increasing and decreasing weighting, general and specific rules, respectively, have more effects on the output to achieve more generalization on test data or precision on learning training data

Expanding the nodes more than necessary can cause overfitting on the training set. This weighting method aggregates general (short) and specific (long) rules to make a good decision. Specifically, if the MT is overfitted on the training data by deep leaves, it can be compensated to some extent by the shallower nodes in the test phase. The shallow nodes capture the general patterns and rules in the data, while the deeper nodes capture more specific and detailed patterns. The proposed ensemble approach aggregates the predictions of all nodes, taking into account their weights, to make a final prediction for a query instance. This aggregation ensures that the contribution of each node to the final prediction is proportional to its importance. In decreasing weighting, overfitting is addressed by ensuring that the weights of deeper nodes are smaller than the weights of shallower nodes.

3.5.2 Single representative tree

One concern with the proposed ensemble approach is the time-complexity of tree construction and reasoning during the training and test phases, respectively. However, the tree construction process in this approach is similar to a simple regression tree, including feature selection, finding splitting points, node expansion, and value or model assignment to the leaves. In contrast to using a regression model in just the leaves (e.g., in M5), the proposed method requires learning a model for all nodes, including the leaves and internal nodes. Each internal node at depth d is treated as a leaf node in a d-structure, and a model is learned on the instances that reach this node. However, all these models are tuned in the offline training phase, rather than during the online testing phase.

One of the main concerns with the proposed ensemble approach is the time-complexity of the reasoning process during the online testing phase. Each query instance must be fed to all the models associated with the met nodes in the path to compute their outputs, which can be time-consuming. To address this issue, the computation can be transferred to the training phase. Specifically, let Q be the set of instances (not limited to the training set) that reach the leaf node $n_D(Q)$. Each instance in Q deterministically meets all and only the ancestors of $n_D(Q)$ in the path from

the root to the leaf node ($n_0(Q),\ldots,n_D(Q)$, where $n_d(Q)$ is the node in depth d that is met by Q).
For all instances in Q, the same weights $w_0,\ldots,w_D$ and models $y_0(q),\ldots,y_D(q)$ are used for output generation. However, $y_d(q_1)$ may not be equal to $y_d(q_2)$ for two different vectors q_1 and q_2 in Q, unless a constant value is assigned to each node. Finally, instead of having $D+1$ different models in the path, it is possible to generate just one model in each leaf by integrating the models in the path using equation (1). It should be noted that even the weights can be adaptively determined.
By using the same weights and models for all instances in Q, the time-complexity of the reasoning process during the online testing phase may be reduced significantly. In some cases, such as in AID and M5 models, the outputs of all nodes are constant or can be represented by a polynomial regression model of degree M. In these cases, the final output can also be constant or a polynomial of the same degree. Therefore, instead of averaging the nodes' output in the path during reasoning, a constant output or polynomial regression model can be set to just the leaves, making the time-complexity of reasoning equal to that of a single regression tree. However, such value or model is not computed using only the target values of instances belonging to the that leaf. Other instances also contribute in this value respect to their roles to determine the values of the nodes in the path and associated weights.
Hence, the problem of stopping condition adjustment changes to value/model assignment to the leaves. In other words, the main difference between the proposed method and the base regression tree is in the value/model used in the leaves for reasoning, with similar time-complexity. This computation can also be used for other regression models, but the weighted average of base models is not necessarily a model of the same type. In this case, the time-complexity may not be reduced.
As the final quality evaluation, the proposed method is compared with general strategies for hyperparameter setting in Table 3. As mentioned, Evolutionary strategies can also be used for tree construction including hyperparameters [32]. But evolutionary models are highly time-consuming such that evaluating each solution needs a tree construction and evaluation. The most usual strategy is grid search where a limited set of values for a hyperparameter (often single parameter) is considered and the best one based on cross-validation is selected. In order to be sure, these values address different and all possible trees, entire path of the parameter may be used in the literature [33]. The entire path of stopping condition parameters can easily be extracted by traversing the Mother Tree; although, based on our knowledge, no research yet has proposed it. Ensemble of models with different parameters is also a good approach which is not indeed a search strategy [34]. However, it uses various models with different parameters (instead of selecting just one value) and aggregates their outputs. One of the contributions of this paper is proposing an ensemble model for defining the stopping condition. The main merit of ensemble models is their capacity to generate outputs by aggregation which are not even seen in each base model. The main drawback of ensemble models is their storage need and inefficiency during reasoning because the query instance should be fed to all base models stored in the memory. But the proposed method can integrate all the models in one structure to leverage merits of ensemble models along with low complexity in time and memory.
One of the most important merits of the proposed model is that, for a query instance can compute the final output with any stopping condition parameter without need to reconstruct the tree using the training set.

4 Experimental results

In this section, the proposed ensemble approach is compared with single stopping conditions in decision trees. Other regression methods are not considered because, it is assumed that decision tree in a specific application is preferred by an expert as a regression model due to their accuracy, speed in reasoning, or interpretability. Random forests may outperform single trees by sacrificing speed of reasoning and interpretability. However, the proposed method, despite being based on an ensemble of trees, can be merged and described by a single tree. As mentioned before, even the resulted tree can be converted to a single tree with just labels or models in the leaves.
In the conducted experiments and comparisons, six datasets presented in Table 4, were used. Two of these datasets, "Stock portfolio performance" and "Concrete Compressive Strength," have two target values for prediction. As a result, eight datasets in the experimental results were used. The proposed ensemble approach was compared with single decision trees using different stopping conditions, such as Maximum Depth of the tree (MxD), Maximum Error per leaf (MxE), and Minimum Relative Promotion in error (MnRP). The experimental results show that the proposed ensemble approach outperforms single stopping conditions.

In this paper, two types of regression trees were implemented based on AID and M5. Each tree construction has two phases: in the first phase, the tree is constructed by selecting the best features and splitting points, while in the second phase, a model is assigned to the leaves. Based on AID, a constant value (mean of the target value of belonging instances) is assigned to each leaf. In M5, although the tree is constructed similarly, a linear regressor is finally assigned to each leaf. In other words, based on constant and fast regressors, the tree is constructed, and finally, a linear regressor is modeled on the instances of each leaf. The use of constant regressors in AID-based trees and linear regressors in M5-based trees allows for flexible modeling of the target variable, depending on the nature of the dataset. In this paper, each feature is partitioned into two branches in an internal node. However, it is permitted to use it again as the pivot feature in descendants, indirectly supporting partitioning to more than two branches.

The proposed trees were tested on all datasets addressed in Table 4, with the only stopping condition being Maximum Depth of tree (MxD). In the first experiment, Mean Square Errors (MSE) of prediction by

both AID and M5 were computed vs. MxD, using five-fold five-times cross-validation. In Fig. 4, for each MxD, the number of datasets for which AID or M5 had better functionality than the other one was reported. For example, for MxD of at most two, M5 defeated AID on all datasets. However, as depth increased up to 20, the number of datasets on which AID achieved better prediction also increased. The main reason for this is that, by increasing the depth of the tree, the complexity of the tree also increases. In such situations, simpler models in the leaves have better generalization on unseen data. The best model for each dataset can often be easily detected by cross-validation. Therefore, to be fair, the best model was used for each dataset (i.e., M5 was used on "Housing" and "Concrete-Slump", while AID was applied on the others).

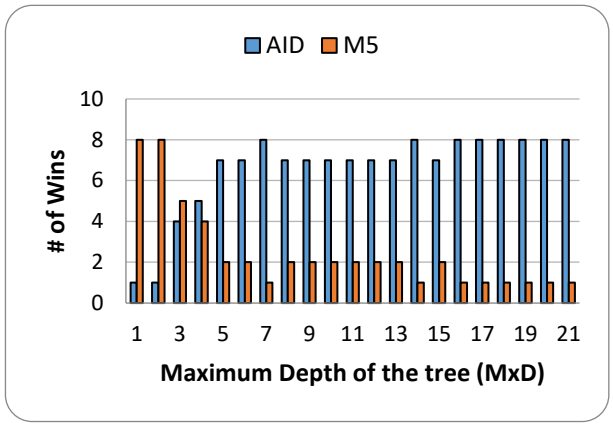

Figure 4: The number of wins of AID (Morgan, 1963) and M5 (Witten, 2005) in competition on the datasets addressed in Table 4 vs. MxD.

Table 5 reports the normalized MSE of predictions on the datasets with different values of MxD. The MSE has been normalized by Mean Square of the target value under prediction, to diminish the scaling effect of target values on the error. The last two rows of the table show the minimum normalized MSE and associated depth for each dataset.

While using maximum depth as a stopping condition has some merits, it also has some drawbacks. On the one hand, this parameter is simple and has a practically finite set of values (positive integers less than an acceptable bound), which makes it easy to implement and tune. On the other hand, this parameter acts similarly in all branches, whereas two nodes at similar depths may have completely different situations with respect to the number of associated instances, variance of target values, and separability of instances in the next expansion. As a consequence, finding the proper value of MxD for a dataset is a challenge, and it may not work properly in all parts of the tree.

In the next experiment, the best regression tree for each dataset (AID/M5) was constructed with another stopping condition, such that one node would not be expanded unless the Sum Square Error (SSE) of associated instances was greater than a predefined threshold. This threshold is referred to as Maximum Error (MxE) in this paper. The results of this experiment are reported in Table 6, with respect to normalized MSE vs. 10 different values of MxE with various granularities. The two last rows of the table present the minimum error and corresponding value of MxE for each dataset.

However, even for this stopping condition, as shown in Table 6, different thresholds can lead to the best performance. For example, the labels of "Stock Portfolio" (Stock Annual & Excess) achieve the best prediction with MxE=0.01, while for "Slump-Slump," the best performance is associated with MxE=100. This stopping condition also suffers from some drawbacks, including its scale sensitivity. If the training set is duplicated, it is not desired to have any modification in the decision tree. However, the number of instances in each node is doubled, and therefore, stopping conditions based on MxE or Minimum Instance per leaf (MnI) may be held in deeper nodes. To address this issue, Mean Squared Error (MSE) of each node may be checked as the stopping criterion instead of SSE. Another scaling problem is related to the scale of the target value. If all target values are scaled by α, it is desired to have a similar tree for reasoning but with scaled values at the leaves. However, Sum Squared Error and MSE of each node are scaled by α^2. Hence defining a threshold is dependent to the scale of the target values.

The third stopping condition investigated in this paper is based on Relative Promotion (RP), which is presented in equation (6).

$$RP(n) = 1 - \frac{\sum_{b=1}^{B} SSE(n_b)}{SSE(n)} \quad (6)$$

Equation (6) defines RP as the difference between the sum of squared errors (SSE) of the parent node and the sum of SSEs of its children, divided by the SSE of the parent node. Here, $SSE(n)$ is the Sum Square Error of associated instances to the node n. The node n is considered to be expanded to B children $n_1, \ldots, n_B$. In other words, this metric measures how the expansion decreases the sum square error of prediction on associated instances, relative to the initial SSE of the parent node. The proposed approach uses RP as a stopping condition, such that a node will not be expanded if its RP is less than a predefined threshold. This threshold is referred to as Minimum Relative Promotion (MnRP) in this paper. Unlike other stopping conditions, such as MxD and MxE, RP is stable with respect to both the training set size and target value scaling. Table 7 shows the normalized MSE of experiments using the proposed stopping condition with different values of MnRP for tree construction on the datasets.

Based on the results, although less oscillation of normalized errors can be seen in comparison with previous metrics, and there is a proper value of MnRP (i.e., 0.2) for most of the datasets, the final results are not sufficiently good. Table 8 compares the best results (minimum normalized MSE) of the stopping conditions in Tables 5-7, which can provide insights into the relative performance of each stopping condition for different datasets.

Table 8 ranks the stopping conditions based on their achieved results for each dataset. Each stopping condition

is assigned a rank equal to the number of other stopping conditions with less error plus one. For example, on "Yacht Dynamics," both MxD and MxE achieved the best rank, and MnRP gets the third rank. The last column of the table shows the average rank of each stopping condition across all datasets. Based on the results, MxD could be the best stopping condition on all datasets except for "Stock Excess," where it achieved the second rank. MxE could achieve the best error on four datasets, whereas it could be better than MxD on just "Stock Excess." However, MxE was the worst stopping condition on other datasets. MnRP could never be the best stopping condition on any dataset, but on average, it achieved a better rank than MxE on these datasets. Due to the stability of results based on MnRP (presented in Table 7), it can be an acceptable stopping condition when the user does not know the best value of the parameter. In the case of having a chance to find out the best parameter, MnRP is better than MxE on average, but cannot compete with MxD.

Based on these experiments, it seems that MxD can be a good stopping metric if there is enough knowledge to find the best value of MxD. However, the proposed model claims that an ensemble of trees with different depths may achieve better results on average without any knowledge requirement. To evaluate this claim, the proposed model has been compared with a set of trees with depths ranging from 0 to 7 in Table 9. The proposed model combined all of these eight trees in a single tree with MxD=7 and five aggregation strategies: Middle, Median, Mean, Incresing weighting, and Decreasing weighting. The methods have been compared based on the resulting normalized MSE.

In addition to normalized MSE, the rank of each method for each dataset has been computed and shown in parentheses. The last columns of the table show the average rank and error of each method across all datasets. The results indicate that ensemble methods significantly outperform fixed depth trees. The worst average error of proposed models (0.1349) is achieved for Middle representative which is better than the best one for single depth models (0.1352 for depth 2). The best average error is 0.1049 for Decreasing Weighting.

Considering the ranks, just on "Housing" and "Yacht" the best method is a single depth model with depths five and six, respectively. On other datasets, the best rank of a single depth model is 4th. Four depths 0-4, no model can achieve a rank better than 5th on any dataset. With any fixed depth, there is at least one dataset for which, the model is ranked 10th or more. Whereas Decreasing Weighting model is the 1th model on four datasets and at most 8th model on others. The Mean model is 2nd or 3rd rank on all datasets except of "Yacht". This is why; the best average rank is achieved for Mean and the second one is Decreasing Weighting. In Fig. 5, average of ranks for fixed depth models on each dataset is compared with the proposed ensemble models. As depicted, ensemble models significantly outperform the fixed depth ones on all datasets except of "Yacht".

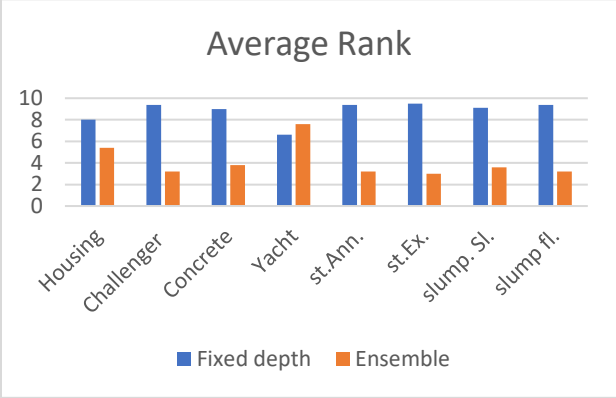

Figure 5: Comparing average of ranks of fixed-depth models with ensemble ones on datasets

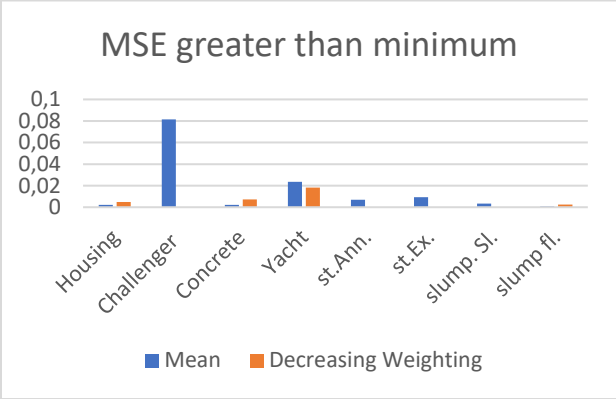

Figure 6: Differences between the normalize MSE of the 1st rank method and each one of "Mean" & "Decreasing Weighting" methods on datasets

In order to have a better comparison between rival proposed methods, "Mean" and "Decreasing Weighting", Fig. 6 is designed to present the difference between the normalized MSE of each method respect to the best one on each dataset. Because the "Mean" method is not the best on any dataset, this difference is always non-zero for "Mean". Although, "Mean" is the 2nd rank on "Challenger", but this difference is significant about 0.08. However, "Decreasing Weighting" method, not only is the best on four datasets, but also has not difference with the best method greater than 0.02.

5 Conclusion

In this paper, a new approach is proposed to ensemble regression trees that are different in maximum depth. The trees are integrated into a single tree using some weighting methods, and all the outputs are merged into a single crisp tree. Based on the experimental results, the proposed methods can achieve better results than even the tree with the best maximum depth on most of datasets, indicating the importance of considering an ensemble of trees with different depths rather than using a fixed stopping condition for all datasets. Moreover, the proposed method achieves significantly better average rank compared to trees constructed up to a fixed maximum depth, with the time complexity of a single crisp tree. Future research

directions may include investigating other reasoning methods, weighting methods, stopping conditions, and making the final function continuous. These extensions can further improve the performance and interpretability of regression trees in various applications.

Acknowledgment

The authors would like to thank Dr. M. Taheri for his help in revising this paper.

References

[1] Charbuty, B., & Abdulazeez, A. (2021). Classification based on decision tree algorithm for machine learning. Journal of Applied Science and Technology Trends, 2(01), 20-28. https://doi.org/10.38094/jastt20165

[2] Nancy, P., Muthurajkumar, S., Ganapathy, S., Kumar, S. S., Selvi, M., & Arputharaj, K. (2020). Intrusion detection using dynamic feature selection and fuzzy temporal decision tree classification for wireless sensor networks. IET Communications, 14(5), 888-895. https://doi.org/10.1049/iet-com.2019.0172

[3] Wang, C., Wang, A., Xu, J., Wang, Q., & Zhou, F. (2020). Outsourced privacy-preserving decision tree classification service over encrypted data. Journal of Information Security and Applications, 53, 102517. https://doi.org/10.1016/j.jisa.2020.102517

[4] Vanfretti, L., & Arava, V. N. (2020). Decision tree-based classification of multiple operating conditions for power system voltage stability assessment. International Journal of Electrical Power & Energy Systems, 123, 106251. https://doi.org/10.1016/j.ijepes.2020.106251

[5] Yang, S. B., & Chen, T. L. (2020). Uncertain decision tree for bank marketing classification. Journal of Computational and Applied Mathematics, 371, 112710. https://doi.org/10.1016/j.cam.2020.112710

[6] Sahoo, S., Subudhi, A., Dash, M., & Sabut, S. (2020). Automatic classification of cardiac arrhythmias based on hybrid features and decision tree algorithm. International Journal of Automation and Computing, 17(4), 551-561. https://doi.org/10.1007/s11633-019-1219-2

[7] Rajesh, B., Vardhan, M. V. S., & Sujihelen, L. (2020, June). Leaf Disease Detection and Classification by Decision Tree. In 2020 4th International Conference on Trends in Electronics and Informatics (ICOEI)(48184) (pp. 705-708). IEEE. https://doi.org/10.1109/icoei48184.2020.9142988

[8] Gomes, C. M. A., Amantes, A., & Jelihovschi, E. G. (2020). Applying the regression tree method to predict students' science achievement. Trends in Psychology, 28(1), 99-117. https://doi.org/10.9788/s43076-019-00002-5

[9] Hu, Y., Dai, Z., & Guldmann, J. M. (2020). Modeling the impact of 2D/3D urban indicators on the urban heat island over different seasons: A boosted regression tree approach. Journal of environmental management, 266, 110424. https://doi.org/10.1016/j.jenvman.2020.110424

[10] Yang, Q., Williamson, A. M., Hasted, A., & Hort, J. (2020). Exploring the relationships between taste phenotypes, genotypes, ethnicity, gender and taste perception using Chi-square and regression tree analysis. Food Quality and Preference, 83, 103928. https://doi.org/10.1016/j.foodqual.2020.103928

[11] Alamgir, M. S. M., Sultana, M. N., & Chang, K. (2020). Link adaptation on an underwater communications network using machine learning algorithms: Boosted regression tree approach. IEEE access, 8, 73957-73971. https://doi.org/10.1109/access.2020.2981973

[12] Shabani, S., Pourghasemi, H. R., & Blaschke, T. (2020). Forest stand susceptibility mapping during harvesting using logistic regression and boosted regression tree machine learning models. Global Ecology and Conservation, 22, e00974. https://doi.org/10.1016/j.gecco.2020.e00974

[13] Avellaneda, F. (2020, April). Efficient inference of optimal decision trees. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 34, No. 04, pp. 3195-3202). https://doi.org/10.1609/aaai.v34i04.5717

[14] Salman Saeed, M., Mustafa, M. W., Sheikh, U. U., Jumani, T. A., Khan, I., Atawneh, S., & Hamadneh, N. N. (2020). An efficient boosted C5. 0 Decision-Tree-Based classification approach for detecting non-technical losses in power utilities. Energies, 13(12), 3242. https://doi.org/10.3390/en13123242

[15] Lotfi, S., Ghasemzadeh, M., Mohsenzadeh, M., & Mirzarezaee, M. (2021). The Construction of Scalable Decision Tree based on Fast Splitting and J-Max Pre-Pruning on Large Datasets. International Journal of Engineering, 34(8). https://doi.org/10.5829/ije.2021.34.08b.01

[16] Panhalkar, A. R., & Doye, D. D. (2021). A novel approach to build accurate and diverse decision tree forest. Evolutionary intelligence, 1-15. https://doi.org/10.1007/s12065-020-00519-0

[17] Muharam, F. M., Nurulhuda, K., Zulkafli, Z., Tarmizi, M. A., Abdullah, A. N. H., Che Hashim, M. F., ... & Ismail, M. R. (2021). UAV-and Random-Forest-AdaBoost (RFA)-Based Estimation of Rice Plant Traits. Agronomy, 11(5), 915. https://doi.org/10.3390/agronomy11050915

[18] Jadhav, D. A. (2021). An enhanced and secured predictive model of Ada-Boost and Random-Forest techniques in HCV detections. Materials Today: Proceedings. https://doi.org/10.1016/j.matpr.2021.05.071

[19] Wang, Q., Zhou, Y., Ding, W., Zhang, Z., Muhammad, K., & Cao, Z. (2020). Random forest with self-paced bootstrap learning in lung cancer prognosis. ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM), 16(1s), 1-12. https://doi.org/10.1145/3345314

[20] Biau, G., & Scornet, E. (2016). A random forest guided tour. Test, 25(2), 197-227. https://doi.org/10.1007/s11749-016-0481-7

[21] Abellan, J., Mantas, C. J., Castellano, J. G., & Moral-Garcia, S. (2018). Increasing diversity in random forest learning algorithm via imprecise probabilities. Expert Systems with Applications, 97, 228-243. https://doi.org/10.1016/j.eswa.2017.12.029

[22] Hornung, R. (2020). Diversity forests: Using split sampling to allow for complex split procedures in random forest. https://doi.org/10.1007/s42979-021-00920-1

[23] Morgan, J. N., & Sonquist, J. A. (1963). Problems in the analysis of survey data, and a proposal. Journal of the American statistical association, 58(302), 415-434. https://doi.org/10.1080/01621459.1963.10500855

[24] Witten, I. H., Frank, E., Hall, M. A., Pal, C. J., & DATA, M. (2005). Practical machine learning tools and techniques. In DATA MINING (Vol. 2, p. 4). https://doi.org/10.1016/b978-0-12-374856-0.00007-9

[25] Fidalgo-Merino, R., & Nunez, M. (2011). Self-adaptive induction of regression trees. IEEE transactions on pattern analysis and machine intelligence, 33(8), 1659-1672. https://doi.org/10.1109/tpami.2011.19

[26] Breiman, Leo. (2017). Classification and regression trees. Routledge. DUA, DHEERU, & GRAFF, CASEY. 2017. UCI Machine Learning Repository. GRISONI, FRANCESCA, CONSONNI, VIVIANA, VIGHI, MARCO, VILLA, SARA, & TODESCHINI, ROBERTO. 2016. Investigating the mechanisms of bioconcentration through QSAR classification trees. Environment international, 88, 198–205. https://doi.org/10.1016/j.envint.2015.12.024

[27] Kordos, M., Piotrowski, J., Bialka, S., Blachnik, M., Golak, S., & Wieczorek, T. (2012, March). Evolutionary optimized forest of regression trees: application in metallurgy. In International Conference on Hybrid Artificial Intelligence Systems (pp. 409-420). Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-28942-2_37

[28] Pham, B. T., Prakash, I., & Bui, D. T. (2018). Spatial prediction of landslides using a hybrid machine learning approach based on random subspace and classification and regression trees. Geomorphology, 303, 256-270. https://doi.org/10.1016/j.geomorph.2017.12.008

[29] Choubin, B., Moradi, E., Golshan, M., Adamowski, J., Sajedi-Hosseini, F., & Mosavi, A. (2019). An ensemble prediction of flood susceptibility using multivariate discriminant analysis, classification and regression trees, and support vector machines. Science of the Total Environment, 651, 2087-2096. https://doi.org/10.1016/j.scitotenv.2018.10.064

[30] Ahmad, M. W., Reynolds, J., & Rezgui, Y. (2018). Predictive modelling for solar thermal energy systems: A comparison of support vector regression, random forest, extra trees and regression trees. Journal of cleaner production, 203, 810-821. https://doi.org/10.1016/j.jclepro.2018.08.207

[31] Zhang, B., Wei, Z., Ren, J., Cheng, Y., & Zheng, Z. (2018). An empirical study on predicting blood pressure using classification and regression trees. IEEE access, 6, 21758-21768. https://doi.org/10.1109/access.2017.2787980

[32] Abdelkader, E. M., Al-Sakkaf, A., Alfalah, G., & Elshaboury, N. (2022). Hybrid Differential Evolution-Based Regression Tree Model for Predicting Downstream Dam Hazard Potential. Sustainability, 14(5), 3013. https://doi.org/10.3390/su14053013

[33] Bak, K. Y. (2023). The regularization paths of total variation-penalized regression splines. Communications in Statistics-Simulation and Computation, 1-12. https://doi.org/10.1080/03610918.2023.2170410

[34] Zhang, Y. L., Shi, Q., Li, M., Yang, X., Li, L., & Zhou, J. (2022). A Classification Based Ensemble Pruning Framework with Multi-metric Consideration. In Intelligent Systems and Applications: Proceedings of the 2021 Intelligent Systems Conference (IntelliSys) Volume 1 (pp. 650-667). Springer International Publishing. https://doi.org/10.1007/978-3-030-82193-7_44

Table 1: Comparing categories of the regression methods

Regression Methods	Interpretability	Storage	Training time	Reasoning time	Performance	Description
Deep	Low ✗	Complex ✗	High ✗	Good ✓	High ✓✓	Need large training data
Neural Network	Low ✗	Good ✓	Good ✓	Low ✓✓	High ✓✓	Local Optima, high capacity of learning
Model Based	Low ✗	Good ✓	Good ✓	Low ✓✓	Good ✓	Global Information
Prototype Based	Good ✓	Complex ✗	Low ✓✓	High ✗	High ✓✓	Local information, storing all prototypes
Rule Based	High ✓✓	Low ✓✓	Good ✓	Good ✓	Good ✓	Need to evaluate by all rules
Random Forest	Good ✓	Good ✓	Good ✓	Good ✓	High ✓✓	Need to evaluate by all trees
Decision Tree	High ✓✓	Low ✓✓	Low ✓✓	Low ✓✓	Good ✓	Interpretable, Compact, Simple, Fast, Accurate

Table 2: Stopping conditions' hyperparameters of regression trees

Stopping Conditions	Merits	Drawbacks
Max. Depth	Simple, Integer, Limited values	Global effect on all nodes, Independent of error
Max. Nodes	Simple, Integer, Limited values, Using storage completely	Needs priority metric, Complex training, Independent of error
Min. Instances	Simple, Integer, Limited values	Sensitive to the size of dataset, Independent of error
Max. Sum Square Error	Considers local errors	Floating point, Unlimited range, sensitive to the size of dataset and the scale of target values
Max. Mean Square Error	Considers local errors, insensitive to the size of dataset	Floating point, Unlimited range, sensitive to the scale of target values, Ignores expandability
Min. Error Promotion	Considers local errors, Considers expandability	Floating point, Unlimited range, sensitive to the scale of target values
Min. Relative error Promotion	Considers local errors, Considers expandability, insensitive to the size of dataset and scale of target values	Floating point, Unlimited range
Information theoretical	Good for classification	Not proper for regression and continuous spaces
Hybrid conditions	More control on tree construction	Complex, multi parameters, Static

Table 3: Search strategies to find hyperparameters of a model

Search strategies	Merits	Drawbacks
Evolutionary Computation: GA, PSO, …	Capable of search in complex spaces including parameters of hybrid conditions	Inefficient: needs to measure the fitness of each solution by tree construction, Static reasoning
Grid Search	Simple	Just one parameter, limited number of candidates: redundancy and missing values, Static reasoning
Entire Path	All possible unique values of parameter	Just one parameter, Static reasoning
Ensemble of models	Generating various outputs	Each parameter set needs a tree, Inefficient reasoning
Proposed model	Generating various outputs, Single tree, Efficient reasoning, testing new conditions without retraining	

Table 4: Datasets used in experiments from UCI Machine Learning Depository (Asuncion, 2007) in regression category.

Dataset	Attributes type	# of attribute	# of rows	labels
Housing	Real	13	506	Housing
Challenger USA Space Shuttle O-Ring	Integer	3	23	Challenger
Concrete Slump Test	Real	10	103	Concrete Slump
Yacht Hydrodynamics	Real	7	308	Yacht Dynamics
Stock Portfolio Performance	Real	12	63	Stock Annual
				Stock Excess
Concrete Compressive Strength	Real	9	1030	Slump _Slump
				Slump Flow

Table 5. Normalized MSE of prediction on datasets vs. MxD

MxD	Housing	Challenger	Concrete Slump	Yacht Dynamics	Stock Annual	Stock Excess	Slump_ Slump	Slump_ Flow
0	0.046	0.844	0.072	0.682	0.051	0.054	0.192	0.113
1	0.064	**0.611**	0.039	0.152	0.045	0.043	0.156	0.082
2	0.059	0.654	0.037	0.035	0.031	0.034	**0.152**	**0.081**
3	0.044	0.751	0.03	0.009	0.03	0.033	0.189	0.09
4	0.037	0.875	0.03	0.006	0.028	0.032	0.204	0.101
5	**0.032**	0.92	0.029	0.006	0.027	0.032	0.215	0.104
6	0.038	0.92	0.029	**0.005**	**0.026**	0.031	0.22	0.106
7	0.034	0.92	**0.027**	0.006	**0.026**	**0.03**	0.22	0.112
8	0.033	0.92	**0.027**	0.006	**0.026**	0.031	0.222	0.114
9	0.034	0.92	**0.027**	0.006	**0.026**	0.031	0.225	0.114
10	0.034	0.92	**0.027**	0.006	**0.026**	0.031	0.225	0.115
Min	**0.032**	**0.611**	**0.027**	**0.005**	**0.026**	**0.03**	**0.152**	**0.081**
Best Depth	5	1	>6	6	>5	7	2	2

Table 6. Normalized MSE vs. 10 values of MxE

MxE	Housing	Challenger	Concrete Slump	Yacht Dynamics	Stock Annual	Stock Excess	Slump-Slump	Slump-Flow
0	**0.034**	0.92	**0.027**	0.006	**0.026**	0.031	0.225	0.115
0.01	**0.034**	0.92	**0.027**	0.006	**0.026**	**0.029**	0.225	0.115
0.05	**0.034**	0.92	**0.027**	0.006	0.028	0.033	0.225	0.115
0.1	**0.034**	0.92	**0.027**	0.006	0.03	0.032	0.225	0.115
0.5	**0.034**	0.92	**0.027**	0.006	0.045	0.043	0.226	0.115
1	**0.034**	**0.775**	**0.027**	0.006	0.051	0.043	0.226	**0.114**
5	**0.034**	0.834	**0.027**	**0.005**	0.051	0.054	0.224	**0.114**
10	0.035	0.844	**0.027**	**0.005**	0.051	0.054	0.226	**0.114**
50	0.045	0.844	0.028	0.007	0.051	0.054	0.217	0.117
100	0.044	0.844	0.028	0.008	0.051	0.054	**0.214**	0.115
Min	**0.034**	**0.775**	**0.027**	**0.005**	**0.026**	**0.029**	**0.214**	**0.114**
Best MxE	[0,5]	1	[0,10]	[5,10]	[0,0.01]	0.01	100	[1,10]

Table 7. Normalized MSE vs. values of Minimum Relative Promotion

MnRP	Housing	Challenger	Concrete Slump	Yacht Dynamics	Stock Annual	Stock Excess	Slump-Slump	Slump-Flow
0	0.034	0.92	**0.027**	**0.006**	**0.026**	0.031	0.225	0.115
0.05	**0.033**	0.92	**0.027**	**0.006**	**0.026**	0.031	0.225	0.115
0.06	**0.033**	0.918	**0.027**	**0.006**	**0.026**	0.031	0.225	0.115
0.07	**0.033**	0.912	**0.027**	**0.006**	**0.026**	0.031	0.226	0.115
0.08	**0.033**	0.912	**0.027**	**0.006**	**0.026**	0.031	0.226	0.115
0.09	**0.033**	0.911	**0.027**	**0.006**	**0.026**	0.031	0.225	0.115
0.1	**0.033**	0.911	**0.027**	**0.006**	**0.026**	0.031	0.217	0.114
0.15	**0.033**	0.716	**0.027**	**0.006**	**0.026**	**0.03**	0.196	0.104
0.2	**0.033**	**0.617**	0.038	**0.006**	**0.026**	**0.03**	**0.183**	0.096
0.3	0.04	**0.617**	0.072	**0.006**	0.039	0.048	0.189	**0.092**
Min	**0.033**	**0.617**	**0.027**	**0.006**	**0.026**	**0.03**	**0.183**	**0.092**
Best MnRP	[0,0.2]	[0.2,0.3]	[0,0.15]	[0,0.3]	[0,0.2]	[0.15,0.2]	0.2	0.3

Table 8. Comparing MxD, MxE and MnRP based on the best Normalized MSE on datasets

Stop. Metric	Housing	Challenger	Concrete Slump	Yacht Dynamics	Stock Annual	Stock Excess	Slump-Slump	Slump-Flow	Avg. Rank
MxD	**0.032 (1)**	**0.611 (1)**	**0.027 (1)**	**0.005 (1)**	**0.026 (1)**	0.03 (2)	**0.152 (1)**	**0.081 (1)**	**1.125**
MxE	0.034 (3)	0.775 (3)	**0.027 (1)**	**0.005 (1)**	**0.026 (1)**	**0.029 (1)**	0.214 (3)	0.114 (3)	2
MnRP	0.033 (2)	0.617 (2)	**0.027 (1)**	0.006 (3)	**0.026 (1)**	0.03 (2)	0.183 (2)	0.092 (2)	1.875

Table 9. Comparing normalized MSE of the proposed model and regression trees with fixed MxD

MxD	Housing	Challenger	Concrete Slump	Yacht Dynamics	Stock Annual	Stock Excess	Slump-Slump	Slump-Flow	Avg. Rank	Avg. Err.
0	0.0462 (11)	0.8443 (9)	0.0723 (13)	0.6824 (13)	0.0508 (13)	0.054 (13)	0.1922 (9)	0.1133 (13)	12	0.257
1	0.0637 (13)	0.6111 (5)	0.0391 (12)	0.152 (12)	0.0451 (12)	0.0429 (12)	0.1558 (6)	0.0817 (7)	9.9	0.149
2	0.0592 (12)	0.6538 (7)	0.0366 (11)	0.0347 (10)	0.0305 (11)	0.0335 (11)	0.1523 (4)	0.0811 (5)	8.9	0.135
3	0.0441 (10)	0.7508 (8)	0.0298 (9)	0.0085 (7)	0.0295 (10)	0.0335 (10)	0.1886 (8)	0.0899 (8)	8.8	0.147
4	0.037 (7)	0.8753 (10)	0.03 (10)	0.0065 (5)	0.0281 (9)	0.0318 (8)	0.2041 (10)	0.1006 (9)	8.5	0.164
5	**0.0322 (1)**	0.92 (11)	0.0293 (7)	0.0056 (3)	0.0271 (8)	0.0324 (9)	0.2147 (11)	0.1036 (10)	7.5	0.171
6	0.0378 (8)	0.92 (12)	0.0289 (6)	**0.0054 (1)**	0.0262 (7)	0.0306 (7)	0.2197 (13)	0.1058 (11)	8.1	0.172
7	0.0338 (2)	0.92 (13)	0.0267 (4)	0.0055 (2)	0.0261 (5)	0.0305 (6)	0.2196 (12)	0.1124 (12)	7	0.172
Middle	0.0359 (5)	0.5874 (4)	0.0288 (5)	0.1506 (11)	0.0254 (4)	0.0271 (4)	0.1481 (2)	**0.0762 (1)**	4.5	0.135
Median	0.0378 (9)	0.5853 (3)	0.0248 (3)	0.0061 (4)	0.0261 (6)	0.0293 (5)	0.1538 (5)	0.081 (4)	4.9	0.118
Mean	0.0345 (3)	0.5723 (2)	0.0245 (2)	0.0288 (9)	0.0243 (3)	0.0259 (2)	0.1491 (3)	0.0768 (2)	**3.3**	0.117
Inc. Weight	0.0347 (4)	0.6164 (6)	**0.0224 (1)**	0.0078 (6)	0.0236 (2)	0.0266 (3)	0.1663 (7)	0.0813 (6)	4.4	0.122
Dec. Weight	0.0369 (6)	**0.4908 (1)**	0.0298 (8)	0.0237 (8)	**0.017 (1)**	**0.0165 (1)**	**0.1456 (1)**	0.0786 (3)	3.6	**0.105**