An Evolutionary Forest for Regression

Hengzhe Zhang, Aimin Zhou, *Senior Member, IEEE*, and Hu Zhang

***Abstract*—Random forest (RF) is a type of ensemble-based machine learning method that has been applied to a variety of machine learning tasks in recent years. This article proposes an evolutionary approach to generate an oblique RF for regression problems. More specifically, our method induces an oblique RF by transforming the original feature space to a new feature space through the evolutionary feature construction method. To speed up the searching process, the proposed method evaluates each set of features based on a decision tree (DT) rather than an RF. In order to obtain an RF, we archive top-performing features and corresponding trees during the search. In this way, both the features and the forest can be constructed simultaneously in a single run. The proposed evolutionary forest is applied to 117 benchmark problems with different characteristics and compared with some state-of-the-art regression methods, including several variants of the RF and gradient boosted DTs (GBDTs). The experimental results suggest that the proposed method outperforms the existing RF and GBDT methods.**

***Index Terms*—Evolutionary feature construction, evolutionary forest (EF), genetic programming (GP), random forest (RF).**

I. Introduction

In RECENT years, decision tree (DT) ensemble methods, such as random forest (RF) [1] and extremely randomized trees (ETs) [2], have been widely applied for regression and classification problems due to their robustness, strong predictive ability, and fast training speed [3]. The ensemble methods depend on a randomized learning process to learn various regressors or classifiers, which may reduce the variance after aggregating the predicted results, and may further improve the predictive performance on the unseen data.

Generally speaking, the performance of ensemble learning methods depends on the diversity and quality of individual predictors [4]. Theoretically, suppose we use mean-squared error as our loss function, the generalization error $E = (F(X) - Y)^2$ of an RF F on an unseen dataset $\{X = \{x_1, \ldots, x_N\}, Y = \{y_1, \ldots, y_N\}\}$ can be decomposed into the following two items [5]:

$$E = \underbrace{\mathbb{E}_f\left[(f(X) - Y)^2\right]}_{\text{average loss}} - \underbrace{\mathbb{E}_f\left[\left(f(X) - \mathbb{E}_{f'}[f'(X)]\right)^2\right]}_{\text{ambiguity}} \quad (1)$$

where f, f' represents all base learners belonging to the RF F. The first item indicates that the generalization performance of an RF can be improved by improving the generalization performance of any base learner, while the second item claims that the generalization performance can also be improved by increasing the diversity among base learners. In the ensemble learning domain, there have been numerous works that concentrate on increasing the diversity of individual predictors [3]. In contrast, enhancing the predictive capability of individual learners, especially improving it from the optimization perspective, has rarely been studied. Considering it directly controls the generalization loss on unseen data, enhancing the generalization capability of individual learners is undoubtedly a critical point that needs to be investigated.

An RF consists of a set of DTs. Generally, there are two ways to improve the learning capability of a single DT. The first is to find a better leaf node, and the second is to find a better nonleaf node. In the first category, a notable work is the piecewise symbolic regression tree [6], which constructs a linear or nonlinear model to replace a constant-leaf node in the DT. The oblique DT [7] and the optimal DT [8] fall into the second category. However, finding an optimal DT is an NP-hard problem [9], and more importantly, it may not be necessary to spend much time on finding an optimal single tree, as previous literature has shown that an RF with many completely randomized DTs can also achieve striking performance [2]. Therefore, we do not intend to address this issue in this article. In contrast to optimal DT research, which focuses on improving the DT performance by searching for optimal splitting points, researchers in the oblique DT field attempt to improve DT performance by replacing the axis-paralleled splitting plane with a more complicated one. The reason behind this choice is that the axis-parallel space partition may not be able to adequately capture the underlying pattern of a complex dataset. Thus, if we do not use an oblique hyperplane to segment the feature space, we may require more hyperplanes to approximate the complicated decision boundary, which may reduce model accuracy and increase model complexity [10]. As a result, many works in recent years have leveraged this idea to improve the performance of conventional DTs, with many promising results, particularly when incorporating this idea into RF [3].

Manuscript received 22 May 2021; revised 5 September 2021 and 8 November 2021; accepted 13 December 2021. Date of publication 20 December 2021; date of current version 1 August 2022. This work was supported in part by the Science and Technology Commission of Shanghai Municipality under Grant 19511120601; in part by the Scientific and Technological Innovation 2030 Major Projects under Grant 2018AAA0100902; and in part by the National Natural Science Foundation of China under Grant 61731009. *(Corresponding author: Aimin Zhou.)*

Hengzhe Zhang and Aimin Zhou are with the School of Computer Science and Technology, Shanghai Institute of AI for Education, East China Normal University, Shanghai 200241, China (e-mail: zhenlingcn@foxmail.com; amzhou@cs.ecnu.edu.cn).

Hu Zhang is with the Science and Technology on Complex System Control and Intelligent Agent Cooperation Laboratory, Beijing Electro-Mechanical Engineering Institute, Beijing 100074, China (e-mail: jxzhangyi1976@126.com).

This article has supplementary material provided by the authors and color versions of one or more figures available at https://doi.org/10.1109/TEVC.2021.3136667.

Digital Object Identifier 10.1109/TEVC.2021.3136667

1089-778X © 2021 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See https://www.ieee.org/publications/rights/index.html for more information.

Authorized licensed use limited to: Universitas Indonesia. Downloaded on November 17,2023 at 02:05:24 UTC from IEEE Xplore. Restrictions apply.

However, even though some advanced RF methods adopt oblique hyperplanes, most of them are still linear [11], which may not be effective in capturing the relationship between variables. Thus, the goal of this article is to find an effective way to improve the performance of DTs by inducing nonlinear hyperplanes during the training process and to improve the performance of RFs by combining those nonlinear oblique DTs. Inspired by the promising results of evolutionary feature construction methods in recent years [12], in this article, we try to investigate whether we can improve the performance of RFs by using evolutionary synthesized features. Based on such an idea, we propose an evolutionary forest (EF), which assembles multiple sets of nonlinear features to form an oblique RF. Specifically, in this article, we use a multitree genetic programming (GP) method to construct nonlinear features, i.e., each GP individual represents a set of features. In order to obtain an RF, we maintain an external archive to store various well-performed GP individuals during the evolutionary process. At the end of evolution, we build a set of DTs based on the stored GP individuals to form the final ensemble model. The main contributions of this article are summarized as follows.

1) A novel approach is proposed to generate RF. Unlike existing RF methods, which rely on completely random transformation, the method proposed in this article focuses on synthesizing expressive features using an evolutionary feature construction method, which helps to improve RF performance. Furthermore, our experimental results show that these constructed features can be used to improve the performance of other cutting-edge tree-based ensemble learning algorithms.
2) Given that directly applying the existing wrapper-based feature construction method to an RF is extremely time-consuming, we instead choose to evaluate each GP individual on a single DT rather than an RF and maintain diversity through a special selection operator. Finally, we combine several individuals to construct an RF. As a result of such a paradigm, our algorithm achieves up to a 48-fold speedup over the traditional wrapper-based feature construction method while maintaining comparable predictive performance.

The remainder of this article is structured as follows. Section II reports the related work. Section III presents the motivation of this work. Section IV introduces the general framework and detailed implementations of EF. Section V empirically studies the behavior of EF based on the framework of bias–variance decomposition. Section VI contains experimental results of EF. Section VII concludes this article and makes some suggestions for future research.

II. Related Work

A. Evolutionary Feature Construction

There has been plenty of work that uses GP methods for feature construction [13]–[15]. One early work uses a tree-based GP method to construct a discriminative feature for each class [16], which attempts to maximize the separability of one class by transforming the original space into another feature space. The results of the experiments show that the evolutionary feature construction method can improve the predictive performance of multiple machine learning classifiers. After that, in [17], the GP method has been successfully used to select and construct compact features on high-dimensional datasets. In [18] and [19], multiple tree GPs have been validated as a promising feature construction scheme in the classification and regression scenarios, respectively.

Although evolutionary feature construction methods have achieved promising results on improving the performance of basic machine learning algorithms, like k-nearest neighbor and DT algorithms [18], it is unclear whether these feature construction methods can be applied to more advanced machine learning algorithms like RF methods. As shown in a recent work [20], even though the GP-based feature construction method can work well on simple machine learning algorithms, its efficiency on complex machine learning algorithms is not satisfactory. In [20], the proposed method requires an hour to evolve a set of features on an RF, even though the experimental dataset only contains 308 items.

B. Oblique Decision Tree

In the oblique RF [21], each individual tree divides the search space by multiple oblique hyperplanes rather than axis-parallel planes. Recently, numerous studies have demonstrated that oblique forest methods outperform traditional RF methods [22]–[24]. In general, because inducing an oblique hyperplane is an NP-Hard problem, we can divide existing methods into three categories based on the hyperplane-inducing method.

1) The first category of methods generates an oblique DT by transforming the original search space into a new feature space, such as by rotating the search space using the PCA method [25]. When all features are continuous, some works claim that rotation forest (RoF) is the best classifier compared to XGBoost or multilayer perceptron (MLP) [26]. However, it is unknown whether such a conclusion is still applicable when categorical features are present.
2) The second category of methods builds an oblique hyperplane using existing classifiers such as SVM. Among 190 classification methods, the best one in this category achieves state-of-the-art performance [11]. Even though such a method can quickly find a local optimal hyperplane for a binary-classification problem at each nonleaf node, finding such a hyperplane for regression and multiclassification problems is still a time-consuming process.
3) The final category of methods converts the oblique RF inducing problem to an optimization problem. The parameters of an oblique DT can be optimized for a given DT structure using stochastic gradient descent [27], [28] or alternative optimization [29] methods. When inducing a single oblique DT, this method performs well. However, there is still no evidence that these tree-inducing methods outperform the traditional RF in the ensemble learning scenario [28].

All of these methods have advantages and disadvantages. Nonetheless, it is worth noting that all of these methods can only produce a linear separation hyperplane. Compared to

Authorized licensed use limited to: Universitas Indonesia. Downloaded on November 17,2023 at 02:05:24 UTC from IEEE Xplore. Restrictions apply.

linearly transforming the feature space, nonlinearly transforming the feature space, such as transforming features by the neural network [30], may be more conducive to achieving better predictive performance. However, although some existing methods produce promising results based on the neural network, they have a significant impact on the interpretability of the oblique RF. As a result, building a nonlinear oblique RF using an evolutionary feature construction method might be a better option for deploying such models in high-risk scenarios.

C. Ensemble Learning in Genetic Programming

The origins of ensemble learning in GP can be traced back to [31], which runs GP procedures multiple times within the boosting and bagging framework. However, such paradigms are time consuming, and it is preferable to take advantage of the population nature of evolutionary algorithms (EAs) to obtain multiple base learners in a single run. In [32], multiple base learners evolve under a spatially structured EA. The results show that evolving multiple base learners at the same time is feasible and can outperform the traditional bagged GP method on several datasets. Following that paper, in [33], some researchers propose a co-EA for performing symbolic regression and ensemble selection tasks at the same time. Moreover, in the GP-based image classification domain, many GP-based ensemble learning algorithms have also been proposed in recent years. For example, in [34], researchers propose to use GP to assemble different image classifiers. Furthermore, in [35], DCFL-FGP is proposed to demonstrate that using a multipopulation scheme to evolve an ensemble of GP classifiers can achieve even better results. However, it should be noted that in [36], a researcher demonstrates that GP is naturally suited to evolve an ensemble model, so both niching method [32], [37] and co-evolutionary method [33] may not be required. Unfortunately, the majority of these papers only investigate the possibility of ensemble GP from the standpoint of EAs rather than feature construction. As a result, although some of them have achieved comparable performance to RF [32], their performance is still inferior to state-of-the-art machine learning methods [33], such as XGBoost [38]. One possible explanation is that the original GP method may have limited modeling capability, e.g., it may be difficult to achieve near-zero training loss when compared to DT or neural network. Consequently, in contrast to these papers, we investigate the ensemble learning of GP in the feature construction scenario in this article. Under this new framework, our goal is to outperform state-of-the-art regression methods.

III. Motivation

To intuitively demonstrate the drawback of existing tree-based ensemble learning methods and evolutionary feature construction methods, we empirically study these two kinds of methods on a curve fitting problem in this section. Let $x \in [-1, 1]$ and $y = x^2 + |x| + \sigma$ be the independent and dependent variables, respectively, where $\sigma \sim \mathcal{N}(0,\ 0.01)$ represents the noise. We uniformly sample 20 data points from $[-1, 1]$ as the training data. In the testing stage, we evenly sample 400 data points from $[-1, 1]$ as the testing data. To have a fair comparison, we run the algorithms ten times with different random seeds and present the mean values and the confidence intervals of the predicted results.

First, we consider tree-based ensemble learning methods. We select four bagging methods and six boosting methods, and the details of these methods are introduced in Section VI-A. The experimental results are shown in Fig. 1. It is clear that the confidence interval of boosting methods is generally large, indicating that these algorithms, particularly LightGBM and DART, are insufficiently stable, i.e., their variances are relatively high. It also suggests that most existing ensemble learning methods are incapable of learning a smooth model, resulting in rugged results. One possible explanation is that all of these methods fit the training data on either the original feature space or a linearly transformed feature space. As a result, these algorithms may struggle to fully depict the relationship between the independent variable and the dependent variable. One possible solution to such a problem, as illustrated in [3], is to design a better feature space. GP might be a suitable choice for resolving this issue, as it has been shown to be promising for feature construction in various scenarios [18]. To verify this hypothesis, we conduct an experiment in which GP is used to create features for a DT model. The results of the experiment are shown in Fig. 1. It is obvious that the constructed features can significantly improve the predictive performance of the DT algorithm. However, it is still inferior to ET in terms of the average predictive performance. Therefore, it is worth exploring how to further improve the performance of existing GP-based feature construction algorithms in the DT scenario.

Intuitively, one possible solution to this problem is to combine multiple constructed features based on the ensemble learning philosophy. In the evolutionary feature construction domain, preserving a group of best constructed features or several groups of Pareto nondominated features [39] is not a novel idea. Nonetheless, in these algorithms, the slightly inferior features will be completely discarded. However, in the machine learning scenario, the best performing features on training data may not be able to perform well on unseen data [40]. Thus, these features may still be useful for prediction on unseen data. In the following example, supposing that we have constructed two features, x^2 and $(\sqrt{x^2+1})^{-1}$, for the given regression problem. Fig. 2 shows that the tree built on $(\sqrt{x^2+1})^{-1}$ works better than the tree built on x^2. However, the tree built with the better feature is incapable of producing an accurate prediction result comparable to ET. In order to solve this problem, we average the prediction results of two trees from Fig. 2(a) and (b) and present the results in Fig. 2(d). It is clear that a forest with two trees outperforms the two individual trees in terms of prediction error. To further confirm that the performance improvement is brought by the ensemble feature construction mechanism, we also show a combined model of two simple random DTs in Fig. 2(c). According to the results, building a forest with two trees in the original feature space is insufficient for fitting the training data.

In summary, the first experiment shows that existing ensemble learning methods may fail to fit a smooth model in some cases, and the traditional GP-based feature construction method is a promising but insufficient solution. In the second experiment, we demonstrate that the ensemble feature construction mechanism can effectively enhance the existing

Authorized licensed use limited to: Universitas Indonesia. Downloaded on November 17,2023 at 02:05:24 UTC from IEEE Xplore. Restrictions apply.

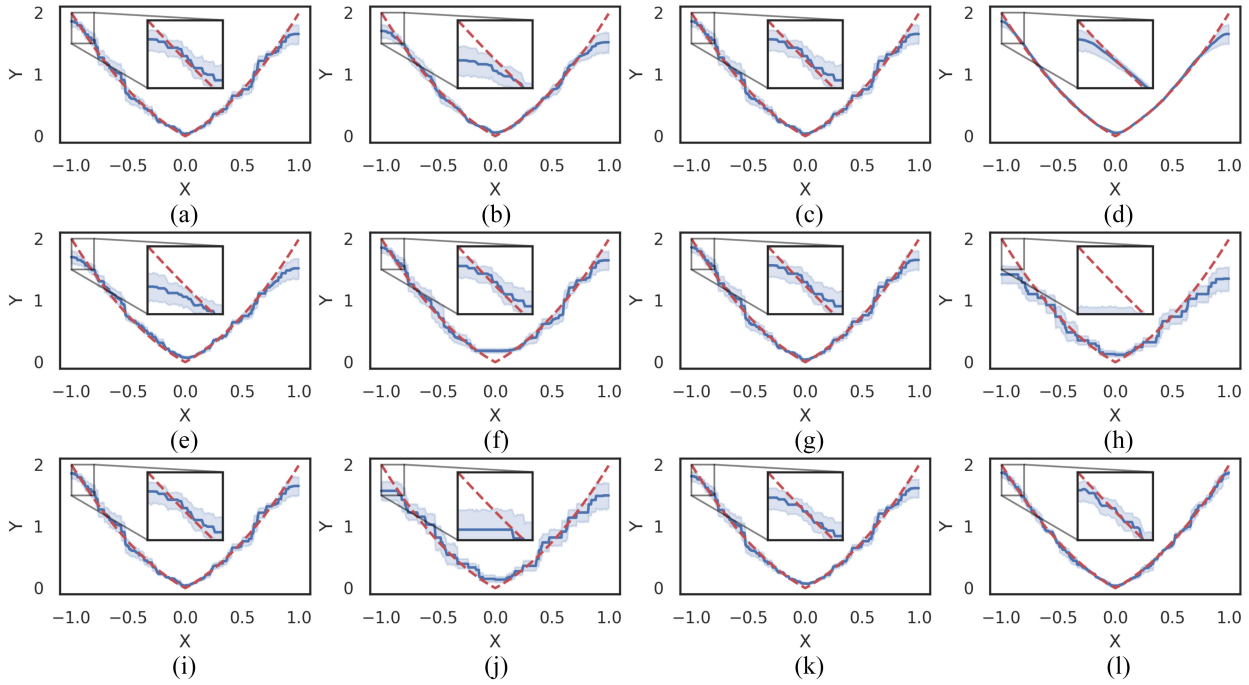


Fig. 1. Predicted results with respect to different learning algorithms on $y = x^2 + |x| + \sigma, \sigma \sim \mathcal{N}(0, 0.01)$. (a) DT Test MSE: 0.02374. (b) RF Test MSEL: 0.02582. (c) RoF Test MSE: 0.02374. (d) ET Test MSE: 0.00813. (e) DeepForest Test MSE: 0.02506. (f) AdaBoost Test MSE: 0.03199. (g) GBDT Test MSE: 0.02375. (h) DART Test MSE: 0.07834. (i) XGBoost Test MSE: 0.02375. (j) LightGBM Test MSE: 0.0673. (k) CatBoost Test MSE: 0.02189. (l) GP+DT Test MSE: 0.01005.

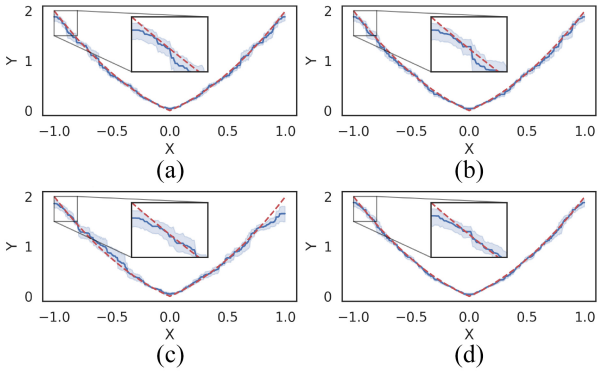


Fig. 2. Predicted results with respect to different features on $y = x^2 + |x| + \sigma, \sigma \sim \mathcal{N}(0, 0.01)$. (a) $(\sqrt{X^2+1})^{-1}$ Test MSE: 0.00972. (b) X^2 Test MSE: 0.0104. (c) X, X Test MSE: 0.02566. (d) $X^2, (\sqrt{X^2+1})^{-1}$ Test MSE: 0.00676.

feature construction method, yielding the best model. Based on the results of the preceding experiments, we can conclude that incorporating nonlinear features into RF might be a promising avenue for obtaining a robust model, and a good way to accomplish this goal is to reserve the relatively poorly performing but diverse features in the evolutionary process to form an ensemble model.

IV. Evolutionary Forest

Following the suggestions in the previous section, this section proposes an EF for regression problems. The algorithm framework, the main operators, and a time complexity analysis are given with details in the following sections.

A. Algorithm Framework

The flowchart of EF is given in Fig. 3. Basically, EF evolves a population of multitree GP individuals, i.e., each individual in the population is a group of GP trees. In each GP individual, multiple GP trees are used to build multiple features. To evaluate a GP individual, a DT is constructed based on that GP individual and the training data. Furthermore, we use the fivefold cross-validation scheme to calculate its fitness value. Then, the GP individuals with good performance are stored in an archive and will form the target forest at the end of the evolutionary process. The pseudocode of EF is presented in Algorithm 1, and some comments are listed as follows.

1) *Initialization:* In lines 1 and 2, a population of GP individuals $P = \{\Phi_1, \Phi_2, \ldots, \Phi_s\}$ are randomly initialized. Each GP individual Φ_i contains m GP trees, which are initialized by the half-and-half method [41], i.e., the half of initialized trees are full while the other half are not. The archive A is set to be empty.
2) *Generation:* In lines 5–9, new candidate GP individuals are generated by applying crossover and mutation operators to parent GP individuals, and the details are given in the following section.
3) *Evaluation:* In lines 10 and 11, the fitness value of each new GP individual is evaluated based on the cross-validation loss of the training data. In order to be consistent with the lexicase selection operator, we keep the original loss vector rather than aggregating them to a scalar value.
4) *Selection:* In line 12, we select s promising GP individuals as the parent GP individuals from the current population P. To improve the diversity of the population,

Authorized licensed use limited to: Universitas Indonesia. Downloaded on November 17,2023 at 02:05:24 UTC from IEEE Xplore. Restrictions apply.

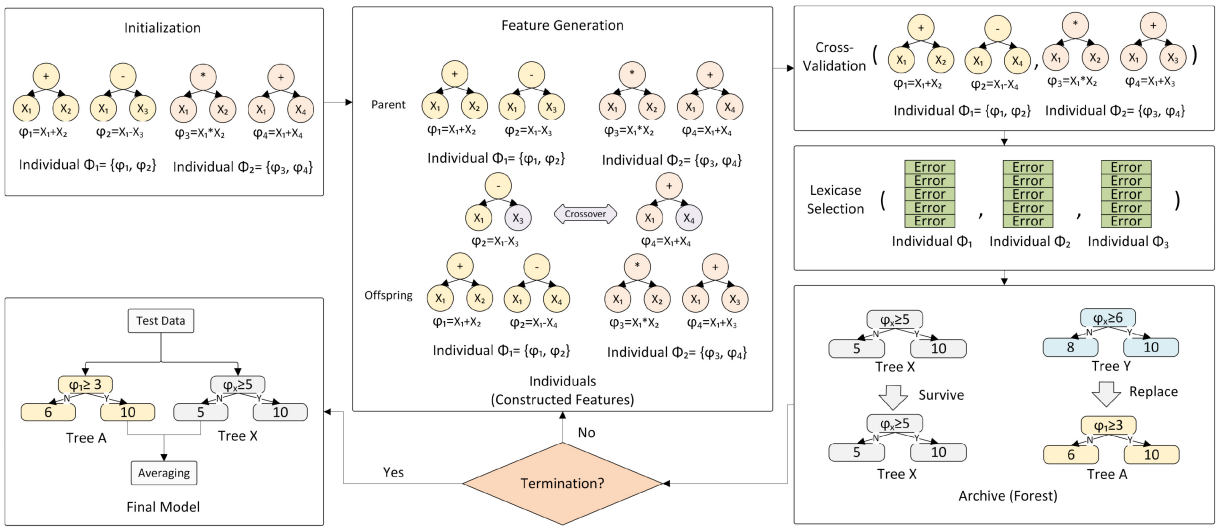


Fig. 3. Illustration of the EF procedure.

Algorithm 1 EF

1: Randomly initialize a population of s GP individuals $P = \{\Phi_1 \dots \Phi_s\}$
2: $A \leftarrow \emptyset$ ▷ Archive Initialization
3: $gen \leftarrow 0$
4: **while** $gen \leq max\_gen$ **do** ▷ Main loop
5: **for** $i = 1, \cdots, s$ **do** ▷ Generation
6: **if** $i\%2 = 1$ & $rand() < p_{cr}$ **then** ▷ Crossover
7: $\{\Phi_i, \Phi_{i+1}\} \leftarrow$ crossover(Φ_i, Φ_{i+1})
8: **if** $rand() < p_{mu}$ **then** ▷ Mutation
9: $\Phi_i \leftarrow$ mutation(Φ_i)
10: **for** $\Phi \in P$ **do** ▷ Evaluation
11: $f_\Phi \leftarrow$ cross validation(Φ)
12: $P' \leftarrow$ selection(P) ▷ Selection
13: $A \leftarrow$ update archive(A, P)
14: $P \leftarrow P'$
15: $gen \leftarrow gen + 1$
return A

we adopt the lexicase selection operator in this article. The details of the selection operators are given as follows.

5) *Archive Update:* In line 13, the new candidate GP individuals are used to update the archive, which shall form the final forest. The details of the archive maintenance method are given as follows.

B. Individual Representation

In this article, we use a GP tree to construct a feature, which is similar to other GP-based feature construction methods [18]. Consequently, a multitree GP in EF naturally represents a set of features that can be used to construct a DT.

Formally, in EF, an individual represents a set of GP trees $\Phi = [\varphi_1, \dots, \varphi_m]$, where φ_i $(i = 1, \dots, m)$ denotes a GP tree. Therefore, each individual also represents an m-D feature set in EF. It should be noted that constructing multiple features simultaneously is becoming popular [18] because such a scheme may alleviate the burden of EAs. More specifically, evolving multiple simple features might be simpler than constructing a complicated feature for achieving the target performance objective.

Based on the above GP representation, for each GP individual, we can build a DT according to the constructed features. In the machine learning domain, a DT can be constructed optimally [9], greedily [42], or randomly [2]. In this article, we use a randomly built DT as our base learner, which means that the splitting point of a DT is chosen at random from all possible options.

An example of EF is illustrated in Fig. 3, where each individual contains two features, and a DT is built solely by leveraging those two constructed features during the tree construction stage. Here, each DT is an oblique DT because it is constructed based on synthesized features. Then, with the EA running, we save well-behaved DTs in the archive and eventually form an EF.

C. Generation Operators

As mentioned above, each individual contains a set of GP trees. In offspring generation, we randomly pick one GP tree from each GP individual to perform mutation and crossover operations as follows.

1) *Subtree mutation* operator randomly chooses a subtree and replaces it with a randomly generated tree.
2) *Subtree crossover* operator randomly replaces a subtree from one parent with a subtree from another parent.

An example of the crossover operation is illustrated in Fig. 3. It should be noted that, to keep population diversity, we record all individuals generated in the past and force the variation operator to generate new individuals.

D. Fitness Evaluation

To evaluate each individual, we employ a fivefold cross-validation scheme to estimate the generalization loss. To this

Authorized licensed use limited to: Universitas Indonesia. Downloaded on November 17,2023 at 02:05:24 UTC from IEEE Xplore. Restrictions apply.

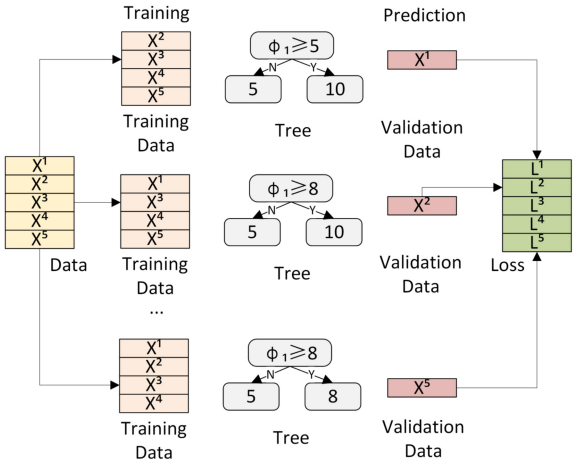


Fig. 4. Fitness evaluation through cross-validation.

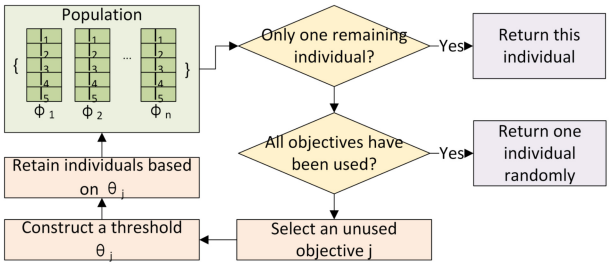


Fig. 5. Illustration of Lexicase selection for choosing one individual.

end, the training set is divided into five equal-sized folds. Among them, four folds are used to train a DT model and the other is used for validation. As shown in Fig. 4, there are five possible combinations. For each combination, a decision model is built and the prediction errors for all the validation data points are recorded. In our implementation, we adopt the absolute deviation as the loss value, i.e., $L_j = |f(X_j) - Y_j|$, where f represents a DT and Y_j represents target values. Instead of averaging all loss values for data items into a scalar fitness value, we concatenate these loss values into a fitness vector $L = [L_1, \ldots, L_5]$ to provide more semantic information to aid in the selection process. It is worth mentioning that if the training set has n data points, then L will have n items because each piece of data point will produce a loss value under the cross-validation scheme. Since we use multiple values to represent the fitness of an individual, we need to design a special selection operator to select the fittest individuals from the population. Thus, in the next section, we will introduce how to design the selection operator.

E. Selection Operator

In the previous section, we introduced that the fitness of an individual in EF is a vector with n items. Thus, we need to design a selection operator to select the fittest individuals based on this vector. The simplest way is to sum this vector into a scalar and then use the traditional tournament selection operator to select the fittest individuals. However, given that diversity is an important factor influencing the performance of an ensemble learning system, we should employ a selection operator that not only prioritizes predictive performance but also encourages diverse predictive behavior. For this reason, we choose the automatic ϵ-lexicase selection operator [43] in this article.

Let $L^i = [l_1^i, \ldots, l_n^i]$ be the fitness vector with the ith individual. The ϵ-lexicase selection works as follows. First, we randomly choose an index $j \in [1, n]$. Then, all individuals with the jth fitness element less than $\theta_j = \min_i l_j^i + \text{mad}_j$ will be preserved, where $\text{mad}_j = \text{median}_i(|l_j^i - \text{median}_k(l_j^k)|)$ represents the median absolute deviation. If more than one individual remains, then we choose another index and repeat the filtering process until there is only one left; otherwise, if there are no indexes left, we randomly choose one. In this way, we choose one individual into the next generation. The procedure is illustrated in Fig. 5. We repeat this procedure until the next population is full-filled.

F. Archive Maintenance

We use an archive to store the found GP individuals. For each newly generated GP individual, if the archive size is less than the predefined capacity, the GP individual will be directly added to the archive; otherwise, the worst GP individual in the archive will be replaced if the new one is better than the worst one. For simplicity, in this step, we average the fitness vector for each GP individual and use this value in the replacement process. Eventually, at the end of the evolution process, each GP individual in the final archive will be used to construct a training set. Then, we train a DT for each constructed training set. In the end, all of these DTs will form a forest, which is the target forest.

G. Time Complexity Analysis

In this section, we examine the time complexity of EF based on the number of DTs constructed in the fitness evaluation process, since it is the most time-consuming part of EF. The reason for this might be that we use a fivefold cross-validation scheme for fitness evaluation and it requires us to build five DTs for each GP individual. Fortunately, in contrast to traditional ensemble methods, the time complexity of EF is not proportional to the ensemble size $|A|$. EF has a time complexity of $O(G|P|)$, where G and $|P|$ represent the maximum generation and population size, respectively. Although EF has a higher time complexity than the traditional boosting method, it has a lower time complexity than the previous wrapped-based feature construction method for the RF [20], whose complexity is $O(G|P||A|)$. For example, if we intend to build a forest with 100 trees, our method will be 100 times faster than other wrapped-based feature construction methods. The empirical study of time complexity is provided in Section VI.

V. Bias–Variance Decomposition

Bias–variance decomposition is a popular analytical tool for identifying bottlenecks in any machine learning system. Traditionally, an error function can be broken down into two parts: 1) bias and 2) variance. The difference between the

Authorized licensed use limited to: Universitas Indonesia. Downloaded on November 17,2023 at 02:05:24 UTC from IEEE Xplore. Restrictions apply.

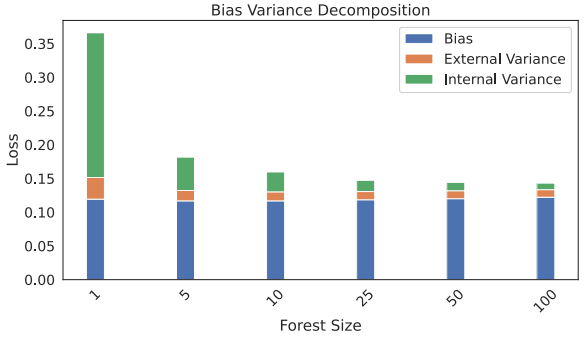


Fig. 6. Bias–variance decomposition with respect to different forest sizes.

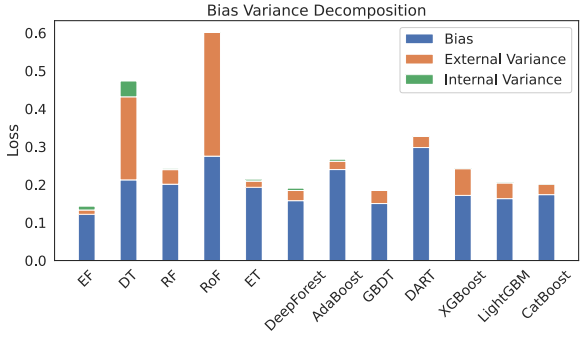


Fig. 7. Bias–variance decomposition with respect to different algorithms.

expected model prediction and real observations is represented by bias, and the average distance between the expected model prediction and each single model value is defined by variance. Recently, as a result of recognizing that the stochastic learning process of some algorithms has a nonnegligible effect, the variance term has been further decomposed into external variance and internal variance [44], which represent the variance induced by the stochastic sampled data and the stochastic training process, respectively. The mean-squared error loss can be decomposed into three components, as shown in

$$\begin{aligned}
&\mathbb{E}_{\mathcal{D},\mathcal{S}}\Big[\{f(\mathbf{x};\mathcal{D},\mathcal{S})-Y\}^2\Big] \\
&\quad=\underbrace{\big\{\mathbb{E}_{\mathcal{D},\mathcal{S}}\big[f(\mathbf{x};\mathcal{D},\mathcal{S})\big]-Y\big\}^2}_{(\text{bias})^2} \\
&\qquad+\underbrace{\mathbb{E}_{\mathcal{D}}\Big[\big\{\mathbb{E}_{\mathcal{S}}\big[f(\mathbf{x};\mathcal{D},\mathcal{S})\mid\mathcal{D}\big]-\mathbb{E}_{\mathcal{D},\mathcal{S}}\big[f(\mathbf{x};\mathcal{D},\mathcal{S})\big]\big\}^2\Big]}_{\text{external variance}} \\
&\qquad+\underbrace{\mathbb{E}_{\mathcal{D},\mathcal{S}}\Big[\big\{f(\mathbf{x};\mathcal{D},\mathcal{S})-\mathbb{E}_{\mathcal{S}}\big[f(\mathbf{x};\mathcal{D},\mathcal{S})\mid\mathcal{D}\big]\big\}^2\Big]}_{\text{interal variance}} \qquad (2)
\end{aligned}$$

where D represents the training sets, and S represents the stochastic training processes. $f(x)$ denotes the predicted result, and Y is the target value. In this article, we follow the guideline proposed in [44] to approximate the decomposed error for each algorithm. For simplicity, we conduct our experiment on 48 datasets with 200 to 500 samples. During the experiment process, we randomly sample ten data subsets and randomly set ten random seeds for each sampled subset to estimate errors and variances due to the limitation of computational resources. For clarity, we normalize the decomposed losses based on the total sum of squares, which is also used in calculating R^2 score, and then present the average losses across all datasets.

In this section, we attempt to analyze the EF character from the evolutionary feature construction domain and the machine learning domain, respectively. Through bias–variance decomposition, we try to answer the following two questions.

1) What is the advantage of ensemble multiple evolved GP individuals?
2) What is the advantage of EF compared to other machine learning algorithms?

The first question seeks to demonstrate the superiority of the proposed method over other feature construction methods used in the GP domain. Fig. 6 depicts the decomposed testing loss for various forest sizes. According to this figure, the traditional feature construction method, which only uses the best GP individual to form the final model, has significantly higher internal variance than the proposed feature construction method. More specifically, as illustrated in this figure, the internal variance of EF with a single tree is greater than that of a forest with five trees. The internal variance has gradually decreased as the forest size increases, while the bias and internal variance have remained constant. Based on the results, we can conclude that the traditional feature construction method is rather fragile and highly dependent on the random seed. In other words, different expressive features may be discovered by the evolutionary feature construction method in different rounds. EF, on the other hand, can greatly reduce this internal variance, resulting in a better generalization capability of the final model.

The second question seeks to ascertain the distinction between the proposed method and other machine learning algorithms, particularly cutting-edge tree-based ensemble methods. Fig. 7 depicts the decomposed testing loss associated with the algorithms. We can see that the single DT method has a high bias, external variance, and internal variance. This means that the traditional DT method is insecure and ineffective at capturing the underlying pattern. When two ensemble paradigms are considered, the results show that the bagged DT methods can significantly reduce the internal variance induced by the DT's random learning procedure, and the boosting method can simultaneously reduce bias and variance. Unfortunately, the majority of these methods continue to have relatively high external variance. This could be because the decision model built on the original feature space is not stable. As a result, even if the input feature space is slightly changed, the constructed model will be greatly varied. As shown in Fig. 7, EF has significantly lower external variance than other ensemble learning methods, indicating that it has discovered robust features and that its prediction will not be greatly affected even if some input data is changed. In addition to this advantage, we can see that the constructed features cause RFs to have a low bias, which also contributes to the superior performance of EF. Thus, to summarize, we show in this experiment that there is still a lot of room for reducing the external variance and bias of current machine learning algorithms, and EF is able to discover some robust features to reduce such variance and bias.

Authorized licensed use limited to: Universitas Indonesia. Downloaded on November 17,2023 at 02:05:24 UTC from IEEE Xplore. Restrictions apply.

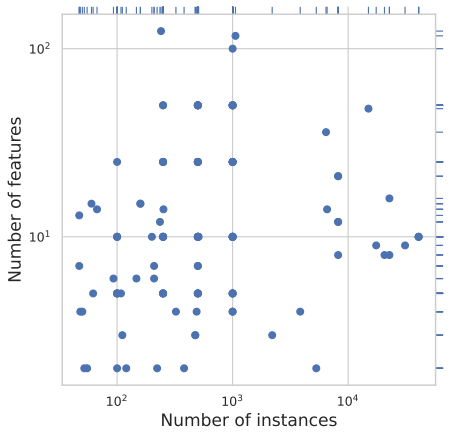


Fig. 8. Properties of the experimental datasets.

TABLE I
PARAMETER SETTINGS FOR GP RUNS

Parameter	Value
Population Size	50
Maximal Number of Generations	100
Crossover and Mutation Rates	GP(0.5 and 0.1)
Maximum Tree Depth	8
Function Set	+,-,*,Analytical Quotient
Forest Size	100
Number of Trees	5

VI. STATISTICAL EXPERIMENTS

This section studies the EF[1] performance through a set of benchmark problems.

A. Experiment Configuration

To validate the effectiveness of EF, we use all datasets within 50 000 data items from the Penn Machine Learning Benchmark (PMLB V1.0) [45] repository. Fig. 8 depicts the number of features and sample sizes for these datasets. In summary, a total of 117 datasets are used in our experiment. The largest dataset has 40 768 items, and the smallest dataset has 47 items. All datasets have the number of features ranging from 2 to 124.

Some state-of-the-art tree-based learning algorithms are chosen as comparison algorithms, which can be divided into two categories. One category refers to bagged tree-based algorithms, including RF [1], RoF [25], ETs [2], and deep forest [46]. A major characteristic with these algorithms is that the building process of individual learners is independent. The other category is boosted tree-based methods, containing adaptive boosting (AdaBoost) [47], gradient boosted DT (GBDT) [38], DART [48], XGBoost [38], LightGBM [49], and CatBoost [50].

Table I summarizes the parameter settings of EF. We use the analytical quotient [51] instead of the protected division because it outperforms the protected division in terms of generalization performance [52]. The random DTs are chosen as base learners in EF. The number of trees in each individual is set to 5. To avoid bias in the parameter settings of benchmark algorithms, we use the default parameters in their open-source implementations. Nonetheless, we fine-tune the hyperparameters of RF, ET, XGBoost, LightGBM, and CatBoost using the halving-success method and the parameter grid provided in [53]. We run 30 independent runs for each algorithm on each instance.

[1]Source code: https://github.com/zhenlingcn/EvolutionaryForest.

TABLE II
COMPARISON OF DIFFERENT FEATURE CONSTRUCTION METHODS ON TESTING LOSS

	EF	RBF-RF
EF	–	92(+)/23(∼)/2(-)
RBF-RF	2(+)/23(∼)/92(-)	–
PCA-RF	3(+)/26(∼)/88(-)	55(+)/44(∼)/18(-)
RP-RF	3(+)/23(∼)/91(-)	33(+)/45(∼)/39(-)
	PCA-RF	**RP-RF**
EF	88(+)/26(∼)/3(-)	91(+)/23(∼)/3(-)
RBF-RF	18(+)/44(∼)/55(-)	39(+)/45(∼)/33(-)
PCA-RF	–	68(+)/42(∼)/7(-)
RP-RF	7(+)/42(∼)/68(-)	–

Unless otherwise stated, the evaluation metric used in this article is mean-squared error, as presented in

$$\text{MSE} = \frac{\sum_{i=1}^{n}(f(x_i) - y_i)^2}{n}. \tag{3}$$

Besides, we define the pairwise Euclidean distance between individual learners as the diversity of each learner similar to [54], which is given in (4) for an individual p

$$\text{Diversity}_p = \frac{\sum_{q=1}^{m}\sqrt{\sum_{i=1}^{n}\left(f_p(x_i) - f_q(x_i)\right)^2}}{m} \tag{4}$$

where m represents the ensemble size.

In the tables, "+," "∼," and "−" in the parenthesis indicate that the performance of the algorithm or parameter in the corresponding row is better than, similar with, or worse than the algorithm or parameter in the corresponding column at a significant level of 0.05 by a Wilcoxon rank-sum test.

B. Effect of Evolutionary Feature Construction

In machine learning, kernel methods and dimensional reduction methods are also widely used for feature construction. In this section, to validate the effectiveness of the evolutionary-based feature construction method, we compare our method with the RBF kernel, principal component analysis (PCA), and random project (RP) methods. Using these feature construction methods, we can transform original features into several constructed features. Following that, multiple ETs are constructed based on these constructed features to form a forest. In general, if we use the same data, PCA will create the same features. Thus, we bootstrap sampled the training data in the PCA learning process to construct diverse features. For the RBF kernel method, we adopt an approximate method for reducing computation time [55]. To have a fair comparison, we fix the ensemble size and use the same base learner. Table II displays the experimental results. It shows that RBF and RP perform similarly, and PCA outperforms them. Nonetheless, on more than two-thirds of datasets, all three methods perform significantly worse than the evolutionary-based feature construction

Authorized licensed use limited to: Universitas Indonesia. Downloaded on November 17,2023 at 02:05:24 UTC from IEEE Xplore. Restrictions apply.

TABLE III
COMPARISON OF DIFFERENT FEATURE CONSTRUCTION PARADIGMS ON TESTING LOSS

	ET	EF
ET	–	11(+)/36(∼)/5(-)
EF	5(+)/36(∼)/11(-)	–

TABLE IV
TRAINING TIME (SECONDS) CONSUMED BY DIFFERENT FEATURE CONSTRUCTION PARADIGMS AND THE SPEEDUP OF EF COMPARED TO ET

	ET	EF	Speedup
Max	4546.62	101.539	48.986
Average	3980.938	84.785	47.02
Min	3600.956	76.063	38.682

method, indicating that the evolutionary-based method outperforms traditional feature construction methods in terms of enhancing the existing RF system.

C. Effect of Feature Construction Paradigm

As mentioned in Section IV, in contrast to the traditional evolutionary feature construction paradigm, our algorithm evaluates a single tree in each iteration and aggregates multiple trees at last to form a forest. Further analysis reveals that the speedup of such a scheme is equivalent to the ensemble size in theory. In this section, we conduct a comparative experiment between the traditional wrapper-based feature construction method (ET) and our proposed method (EF). In the study, the base learner of the traditional algorithm is ETs with 100 individual trees, and correspondingly the archive size of the traditional algorithm is 1. Since such a scheme takes much more time than EF, we only conduct experiments on datasets with less than 500 instances. Table III displays the difference in testing loss. The experimental results show that there is no significant difference between the two methods, confirming that it is reasonable to replace the traditional feature construction method with the proposed method. Table IV shows that the training time cost by the traditional method is usually more than 1 h on average, even though all datasets have no more than 500 instances. In comparison, our proposed method costs no more than 2 min on average, and the average speedup is up to 47 times, indicating the effectiveness of our proposed method.

D. Influence of Control Parameters

1) Effect of Ensemble Size: As stated in Section V, ensemble size is a critical factor in determining the bias–variance tradeoff in EF. Fig. 9 presents testing loss with respect to different ensemble sizes. This figure shows that as the ensemble size increases, the testing loss of EF decreases on these datasets. To investigate the cause of this phenomenon, we plot the archive diversity with respect to different ensemble sizes in Fig. 10. We can see that the archive diversity gradually improves as the ensemble size increases. Since more diverse

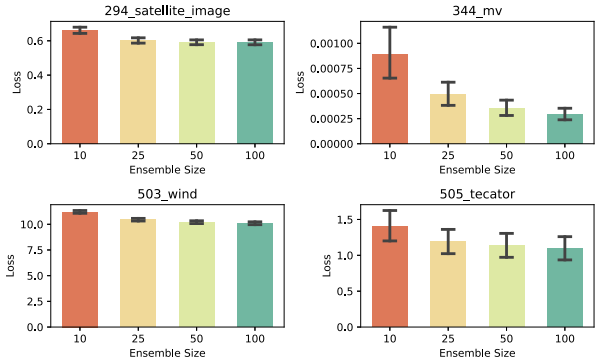


Fig. 9. Testing loss corresponds to different ensemble sizes.

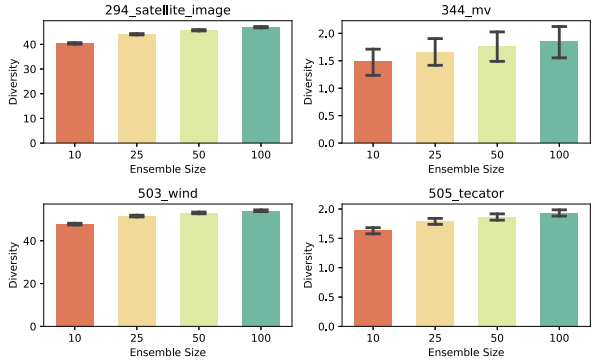


Fig. 10. Archive diversity corresponds to different ensemble sizes.

TABLE V
STATISTICAL TEST OF TESTING LOSSES WITH RESPECT TO DIFFERENT ENSEMBLE SIZES

	10	25
10	–	0(+)/77(∼)/40(-)
25	40(+)/77(∼)/0(-)	–
50	54(+)/63(∼)/0(-)	4(+)/113(∼)/0(-)
100	58(+)/59(∼)/0(-)	11(+)/106(∼)/0(-)
	50	**100**
10	0(+)/63(∼)/54(-)	0(+)/59(∼)/58(-)
25	0(+)/113(∼)/4(-)	0(+)/106(∼)/11(-)
50	–	0(+)/116(∼)/1(-)
100	1(+)/116(∼)/0(-)	–

DTs usually lead to an RF with a better generalization capability [1], it is no surprise that EF can achieve better predictive performance. Table V displays the more detailed experimental results for all datasets. The results on all datasets are consistent with the previously discovered pattern, i.e., the testing loss of EF decreases gradually as the ensemble size increases. Furthermore, the experimental results show that the decrease in testing loss gradually decreases with increasing ensemble size. More specifically, there is a significant difference in testing loss between the ensemble sizes of 10 and 25. However, the difference in testing loss between ensemble sizes of 50 and 100 is not significant. Thus, while a larger ensemble size has a positive impact on testing loss, it may reach a tipping point beyond which the testing loss cannot be improved further even if the ensemble size is increased further. In this article,

Authorized licensed use limited to: Universitas Indonesia. Downloaded on November 17,2023 at 02:05:24 UTC from IEEE Xplore. Restrictions apply.

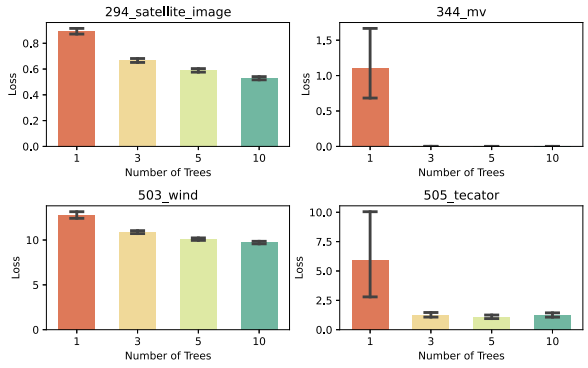


Fig. 11. Testing loss corresponds to different number of trees.

TABLE VI
STATISTICAL TEST OF TRAINING LOSSES WITH RESPECT TO DIFFERENT NUMBER OF TREES

	1	3
1	–	0(+)/43(∼)/74(-)
3	74(+)/43(∼)/0(-)	–
5	78(+)/39(∼)/0(-)	24(+)/93(∼)/0(-)
10	79(+)/38(∼)/0(-)	31(+)/86(∼)/0(-)
	5	**10**
1	0(+)/39(∼)/78(-)	0(+)/38(∼)/79(-)
3	0(+)/93(∼)/24(-)	0(+)/86(∼)/31(-)
5	–	0(+)/98(∼)/19(-)
10	19(+)/98(∼)/0(-)	–

we choose 100 as the default ensemble size since the improvement of bigger ensemble sizes may be negligible as illustrated in Table V.

2) Effect of Number of Trees: In this article, we employ a multitree GP paradigm. Although it may lead to additional computational and memory costs, it brings some advantages. In Fig. 11, we draw testing loss to demonstrate this. Tables VI and VII show the training loss and testing loss, respectively, with different number of trees. It clearly shows that the multitree paradigm always outperforms the single-tree paradigm, despite the fact that there is no optimal tree number. Table VI shows that increasing just two trees for each individual can significantly reduce training loss on 74 datasets. The reason might be that a single feature may not be sufficient to distinguish between different training instances. Thus, including more than one feature in the GP individual will help to improve the discriminatory capability of each DT, potentially improving the generalization performance of the entire forest. Table VII depicts the testing loss for various numbers of trees across all datasets. The trend of testing and training loss, which corresponds to different numbers of trees, is similar. It should be noted that having more trees may not always imply better performance. More trees lead to a larger search space, implying that more computational resources are required to achieve a satisfactory result. For this reason, in this article, we chose 5 as the default number of trees.

E. Contributions of Algorithm Components

1) Effect of Base Learner: Despite the fact that EF uses random DTs as the base learner, it can be applied to a variety of

TABLE VII
STATISTICAL TEST OF TESTING LOSSES WITH RESPECT TO DIFFERENT NUMBER OF TREES

	1	3
1	–	0(+)/32(∼)/85(-)
3	85(+)/32(∼)/0(-)	–
5	86(+)/31(∼)/0(-)	42(+)/75(∼)/0(-)
10	86(+)/29(∼)/2(-)	51(+)/63(∼)/3(-)
	5	**10**
1	0(+)/31(∼)/86(-)	2(+)/29(∼)/86(-)
3	0(+)/75(∼)/42(-)	3(+)/63(∼)/51(-)
5	–	3(+)/102(∼)/12(-)
10	12(+)/102(∼)/3(-)	–

TABLE VIII
STATISTICAL TEST OF TESTING LOSSES WITH RESPECT TO DIFFERENT BASE LEARNERS

	DT	Random-DT
DT	–	5(+)/67(∼)/45(-)
Random-DT	45(+)/67(∼)/5(-)	–

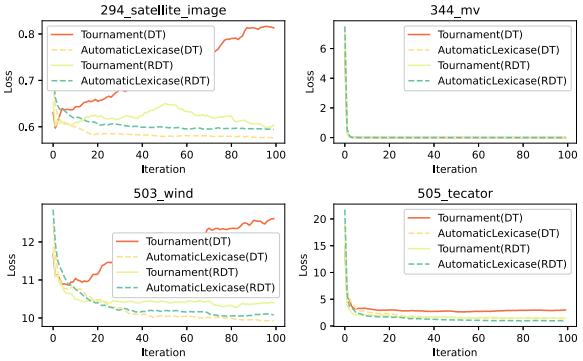


Fig. 12. Testing loss versus iteration for different selection operators.

base learners. This section studies the impact of different base learners. In particular, we compare the predictive performance of EF with DTs or random DTs as base learners. Table VIII presents the experimental results of different base learners. EF with random trees (RDTs) as base learners is preferable to EF with ordinary DTs as base learners, since the former has significantly better performance on 41 datasets while only having degraded performance on four datasets compared to the latter.

2) Effect of Selection Operator: In this section, we investigate the performance effect of different selection operators. We choose the tournament selection operator with a tournament size of 25 as the comparison operator. We use RDTs or ordinary DTs as base learners. Figs. 12 and 13 present the curves of testing loss and archive diversity with regard to four datasets, respectively. Fig. 12 shows the discrepancy between different selection operators. It is clear that, on the "satellite image" dataset and the "wind" dataset, when using ordinary DTs as base learners, EF with the lexicase selection operator reduces the testing loss gradually with the increase of iteration, while EF with the tournament selection operator does not. With the evolutionary process, the testing loss of EF with the tournament selection operator even increases on the satellite image dataset and the wind dataset. Fig. 13 shows the behavioral diversity

Authorized licensed use limited to: Universitas Indonesia. Downloaded on November 17,2023 at 02:05:24 UTC from IEEE Xplore. Restrictions apply.

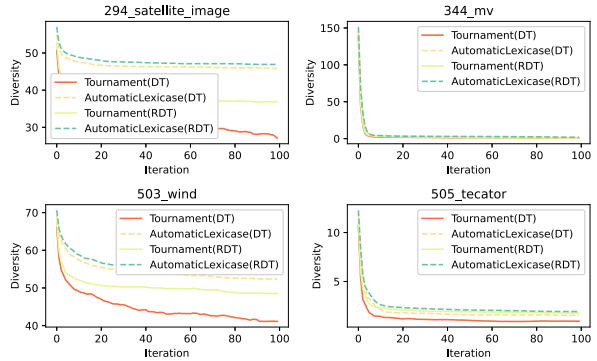


Fig. 13. Archive diversity versus iteration for different selection operators.

TABLE IX
STATISTICAL TEST OF TESTING LOSSES WITH RESPECT TO DIFFERENT SELECTION OPERATORS AND DIFFERENT BASE LEARNERS (USING THE LEXICASE SELECTION OPERATOR AS BASELINE)

	Tournament-3	Tournament-25
DT	53(+)/63(∼)/1(-)	98(+)/18(∼)/1(-)
Random-DT	–	8(+)/88(∼)/21(-)

of individual learners in the ensemble in relation to the two operators. In the later stages, EF with the lexicase selection operator clearly maintains a higher level of semantic diversity than EF with the tournament selection operator. However, when considering the random DT, the situation is very different. It suggests that the diversity of individual learners remains reasonable for both the tournament selection operator and the lexicase selection operator. As a result, when using random DTs as base learners, using a specifically designed selection operator to facilitate semantic diversity may not be necessary. To further demonstrate the effectiveness of the lexicase selection operator, we run experiments on all datasets with three types of selection operators and present the statistical comparison results in Table IX. Specifically, we use the lexicase selection operator as the baseline selection operator and compare it with the tournament selection operator with a size of 3 and 25, respectively. From this table, we find that when using ordinary DTs as base learners, EF with the lexicase selection operator outperforms EF with these two types of tournament selection operators on 53 and 98 datasets, respectively. Nonetheless, when adopting random DTs as base learners, EF with the tournament selection operator is slightly better.

In summary, experimental results demonstrate that when the base learning algorithm cannot provide enough diversity, the selection operator has a significant impact on the performance of EF, and the lexicase selection operator is a good choice for this algorithm. In other cases, emphasizing semantic diversity in the selection operator may not be required.

F. Comparison With State-of-the-Art Methods

1) Comparison With GP Methods: In this section, we compare EF with existing GP-based feature construction and ensemble learning methods. In terms of the GP-based feature construction method, we compare our algorithm to feature engineering wrapper (FEW) [56], which is a GP-based feature construction method that can be used for both regression and classification problems. FEW is a hybrid of filter-based and embedded feature construction methods because it evaluates the quality of each feature using both the feature importance value and the coefficient of determination. As for the GP-based ensemble learning method, we compare our algorithm with the state-of-the-art GP-based ensemble learning method, 2SEGP [36], to demonstrate the superiority of our algorithm.

In terms of hyperparameter settings for FEW, we use the same generation number as EF and a population size five times larger than EF. To ensure a fair comparison, we let FEW construct features for an RF with 100 random DTs. As for 2SEGP, we use the 2SEGP parameter settings directly from its original paper [36] for simplicity. However, because the maximum number of generations of 2SEGP in [36] is the same as that of EF, and the population size used by 2SEGP is up to 500, which is ten times larger than EF, we only run the experiment on 52 PMLB datasets with fewer than 500 instances.

We present a statistical comparison of training and testing losses in Table X. According to the experimental results, even though FEW outperforms EF in terms of training loss, on the testing data, EF outperforms FEW on 60 datasets while only being surpassed on 12 datasets. One possible explanation for this phenomenon is that the features discovered by FEW do not depict a stable relationship between the independent and dependent variables, i.e., these features can only work well on training data, resulting in the overfitting phenomenon. In terms of testing accuracy of 2SEGP, it is clear that EF outperforms 2SEGP on more than half of the datasets. We also present the experimental results with respect to the training losses in Table X to explain why 2SEGP cannot compete with EF. Based on these results, it is clear that 2SEGP does not perform well on training data. In practice, an unpruned DT algorithm is always more likely than a GP model to achieve a near-zero training loss. Thus, such results are not surprising

TABLE X
STATISTICAL TEST OF TRAINING AND TESTING LOSSES WITH RESPECT TO DIFFERENT GP-BASED FEATURE CONSTRUCTION AND ENSEMBLE LEARNING ALGORITHMS (USING EF AS BASELINE)

	Training Loss	Testing Loss
FEW	28(+)/51(∼)/38(-)	60(+)/45(∼)/12(-)
2SEGP	49(+)/3(∼)/0(-)	27(+)/22(∼)/3(-)

TABLE XI
NUMERICAL RESULTS OF DIFFERENT REGRESSION ALGORITHMS

	Mean R^2	Median R^2
EF	**0.836**	**0.921**
DT	0.590	0.650
RF	0.717	0.771
RoF	0.544	0.581
ET	0.726	0.786
DeepForest	0.791	0.840
AdaBoost	0.705	0.761
GBDT	0.799	0.875
DART	0.650	0.766
XGBoost	0.785	0.856
LightGBM	0.659	0.837
CatBoost	0.814	0.877

Authorized licensed use limited to: Universitas Indonesia. Downloaded on November 17,2023 at 02:05:24 UTC from IEEE Xplore. Restrictions apply.

TABLE XII
STATISTICAL COMPARISON BETWEEN DIFFERENT REGRESSION ALGORITHMS

	EF	DT	RF	RoF	ET	DeepForest
EF	–	108(+)/8(∼)/1(-)	86(+)/24(∼)/7(-)	111(+)/5(∼)/1(-)	86(+)/20(∼)/11(-)	78(+)/27(∼)/12(-)
DT	1(+)/8(∼)/108(-)	–	6(+)/19(∼)/92(-)	44(+)/57(∼)/16(-)	7(+)/23(∼)/87(-)	6(+)/7(∼)/104(-)
RF	7(+)/24(∼)/86(-)	92(+)/19(∼)/6(-)	–	94(+)/21(∼)/2(-)	6(+)/74(∼)/37(-)	4(+)/35(∼)/78(-)
RoF	1(+)/5(∼)/111(-)	16(+)/57(∼)/44(-)	2(+)/21(∼)/94(-)	–	3(+)/23(∼)/91(-)	4(+)/4(∼)/109(-)
ET	11(+)/20(∼)/86(-)	87(+)/23(∼)/7(-)	37(+)/74(∼)/6(-)	91(+)/23(∼)/3(-)	–	17(+)/43(∼)/57(-)
DeepForest	12(+)/27(∼)/78(-)	104(+)/7(∼)/6(-)	78(+)/35(∼)/4(-)	109(+)/4(∼)/4(-)	57(+)/43(∼)/17(-)	–
AdaBoost	0(+)/19(∼)/98(-)	83(+)/17(∼)/17(-)	23(+)/44(∼)/50(-)	87(+)/16(∼)/14(-)	20(+)/44(∼)/53(-)	4(+)/29(∼)/84(-)
GBDT	6(+)/38(∼)/73(-)	106(+)/9(∼)/2(-)	80(+)/28(∼)/9(-)	108(+)/8(∼)/1(-)	63(+)/34(∼)/20(-)	42(+)/45(∼)/30(-)
DART	3(+)/9(∼)/105(-)	80(+)/11(∼)/26(-)	50(+)/29(∼)/38(-)	85(+)/9(∼)/23(-)	30(+)/38(∼)/49(-)	19(+)/25(∼)/73(-)
XGBoost	13(+)/33(∼)/71(-)	107(+)/7(∼)/3(-)	81(+)/35(∼)/1(-)	106(+)/10(∼)/1(-)	64(+)/48(∼)/5(-)	39(+)/65(∼)/13(-)
LightGBM	12(+)/18(∼)/87(-)	86(+)/9(∼)/22(-)	70(+)/19(∼)/28(-)	87(+)/13(∼)/17(-)	49(+)/34(∼)/34(-)	43(+)/29(∼)/45(-)
CatBoost	20(+)/41(∼)/56(-)	108(+)/8(∼)/1(-)	91(+)/25(∼)/1(-)	110(+)/7(∼)/0(-)	89(+)/25(∼)/3(-)	73(+)/39(∼)/5(-)

	AdaBoost	GBDT	DART	XGBoost	LightGBM	CatBoost
EF	98(+)/19(∼)/0(-)	73(+)/38(∼)/6(-)	105(+)/9(∼)/3(-)	71(+)/33(∼)/13(-)	87(+)/18(∼)/12(-)	56(+)/41(∼)/20(-)
DT	17(+)/17(∼)/83(-)	2(+)/9(∼)/106(-)	26(+)/11(∼)/80(-)	3(+)/7(∼)/107(-)	22(+)/9(∼)/86(-)	1(+)/8(∼)/108(-)
RF	50(+)/44(∼)/23(-)	9(+)/28(∼)/80(-)	38(+)/29(∼)/50(-)	1(+)/35(∼)/81(-)	28(+)/19(∼)/70(-)	1(+)/25(∼)/91(-)
RoF	14(+)/16(∼)/87(-)	1(+)/8(∼)/108(-)	23(+)/9(∼)/85(-)	1(+)/10(∼)/106(-)	17(+)/13(∼)/87(-)	0(+)/7(∼)/110(-)
ET	53(+)/44(∼)/20(-)	20(+)/34(∼)/63(-)	49(+)/38(∼)/30(-)	5(+)/48(∼)/64(-)	34(+)/34(∼)/49(-)	3(+)/25(∼)/89(-)
DeepForest	84(+)/29(∼)/4(-)	30(+)/45(∼)/42(-)	73(+)/25(∼)/19(-)	13(+)/65(∼)/39(-)	45(+)/29(∼)/43(-)	5(+)/39(∼)/73(-)
AdaBoost	–	2(+)/20(∼)/95(-)	35(+)/16(∼)/66(-)	3(+)/25(∼)/89(-)	27(+)/13(∼)/77(-)	3(+)/26(∼)/88(-)
GBDT	95(+)/20(∼)/2(-)	–	101(+)/12(∼)/4(-)	18(+)/81(∼)/18(-)	44(+)/53(∼)/20(-)	7(+)/47(∼)/63(-)
DART	66(+)/16(∼)/35(-)	4(+)/12(∼)/101(-)	–	2(+)/34(∼)/81(-)	17(+)/25(∼)/75(-)	0(+)/16(∼)/101(-)
XGBoost	89(+)/25(∼)/3(-)	18(+)/81(∼)/18(-)	81(+)/34(∼)/2(-)	–	43(+)/65(∼)/9(-)	5(+)/55(∼)/57(-)
LightGBM	77(+)/13(∼)/27(-)	20(+)/53(∼)/44(-)	75(+)/25(∼)/17(-)	9(+)/65(∼)/43(-)	–	2(+)/28(∼)/87(-)
CatBoost	88(+)/26(∼)/3(-)	63(+)/47(∼)/7(-)	101(+)/16(∼)/0(-)	57(+)/55(∼)/5(-)	87(+)/28(∼)/2(-)	–

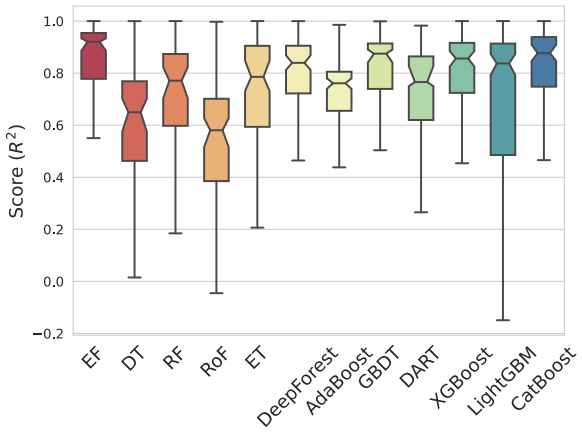


Fig. 14. R^2 performance for EF in comparison to other machine learning algorithms.

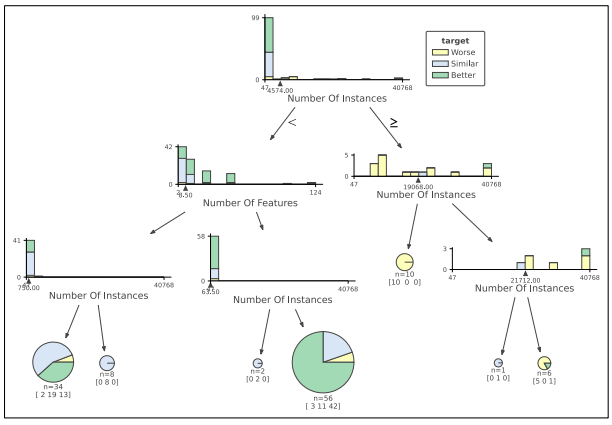


Fig. 15. DT for determining the relationship between EF and CatBoost based on dataset properties.

given that 2SEGP directly employs the GP method to construct a regressor, whereas EF employs the GP technique to construct features for a DT. Finally, based on the above analysis, we can conclude that our DT-based ensemble learning method, EF, outperforms both the traditional GP-based feature construction and ensemble learning methods.

2) Comparison With Tree-Based Methods: In this section, we compare EF with 11 tree-based algorithms, including four bagged tree-based algorithms and six boosted tree-based algorithms. To obtain better results, in this section, we increase the population size to 100. Fig. 14 depicts a general overview of the experimental results for these algorithms, and the corresponding numerical results are presented in Table XI. Without EF, it is clear from this figure that boosted DT methods outperform bagged learning methods in general. The details of the performance discrepancy between the different algorithms are presented in Table XII. Among bagged algorithms, except EF, ETs achieve the best performance, and deep forest performs slightly worse than it. For all of the bagged models, EF significantly outperforms them on more than half of the datasets. As a result, EF is a superior bagging algorithm in this regard. As for boosted algorithms, the experimental results reveal that CatBoost is the best among six boosted DT algorithms. When taking boosted-based learners, which are often to be considered as state-of-the-art algorithms in the classical machine learning field, into consideration, the proposed method is still competitive. More precisely, among 117 benchmark datasets, EF outperforms CatBoost on 56 datasets, while it has been surpassed by CatBoost on 20 datasets.

To better understand the experimental results, we build a DT based on dataset properties and the win–loss relationship between CatBoost and EF on these 117 datasets. Fig. 15

Authorized licensed use limited to: Universitas Indonesia. Downloaded on November 17,2023 at 02:05:24 UTC from IEEE Xplore. Restrictions apply.

depicts the situation. In this figure, "better" means that EF performs better than CatBoost on these datasets, "similar" means that there is no significant difference between the two algorithms, and "worse" means that CatBoost outperforms EF. Based on this figure, we can see that EF and CatBoost perform similarly or better when the dataset contains fewer than 4574 data instances. When the number of features in a dataset is less than or equal to 8, two algorithms perform similarly on more than half of the datasets. This could be that EF is unable to demonstrate the advantage of high-order feature crossing when the number of features in a dataset is insufficient. When there are more than eight features in a dataset and the dataset has fewer than 4574 data instances, EF generally outperforms CatBoost, indicating that EF is a better algorithm when the data is scarce. However, when a dataset contains more than 4574 data instances, EF performs relatively poorly. As a result, in these cases, CatBoost may be preferable because it has better overall performance.

G. Performance Enhancement of Constructed Features

In the previous section, we demonstrated that EF outperforms state-of-the-art tree-based machine learning algorithms on 56 datasets. However, in real-world applications, it may be difficult for practitioners to immediately integrate a new machine learning package into their online systems due to concerns about the new software's potential risks. A feasible solution to the problem would be for the algorithm to generate some transparent features rather than a complex model. Then, during the data preprocessing stage, practitioners can scrutinize and incorporate these features into their existing system.

Fortunately, in addition to inducing a powerful model, EF also contains a plethora of powerful features at the end of the training. Additionally, each DT can calculate the importance of each feature that is used. Consequently, it is possible to investigate whether these newly constructed features can be used to directly improve other state-of-the-art algorithms. To keep things simple, we sum the feature importance values for each distinct feature in EF; i.e., if a feature is used in multiple DTs, it is highly likely that it will have a higher aggregated feature importance value. Then, using the rank of their importance values, we select the first half features and apply them to existing algorithms. For the sake of simplicity, we use the default parameters of each algorithm package. Table XIII and Fig. 16 contain the experimental results. It is clear that EF is also a promising method for improving the performance of existing machine learning systems, as the constructed features outperform existing features on more than half of experimental datasets. Furthermore, it is worth noting that the ensemble feature construction mechanism plays a significant role in achieving such results, i.e., extracting constructed features from the best individual rather than the archive cannot achieve comparable results. The ablation studies are available in the supplementary material, and the experimental results confirm that our discovery is critical for improving the performance of XGBoost or LightGBM.

TABLE XIII
STATISTICAL TEST OF R^2 SCORE WITH RESPECT TO USING CONSTRUCTED FEATURES OR ORIGINAL FEATURES

	+	∼	-
DT	79	36	2
RF	77	38	2
ET	67	43	7
AdaBoost	83	33	1
GBDT	69	44	4
DART	71	43	3
XGBoost	67	44	6
LightGBM	74	36	7
CatBoost	63	44	10

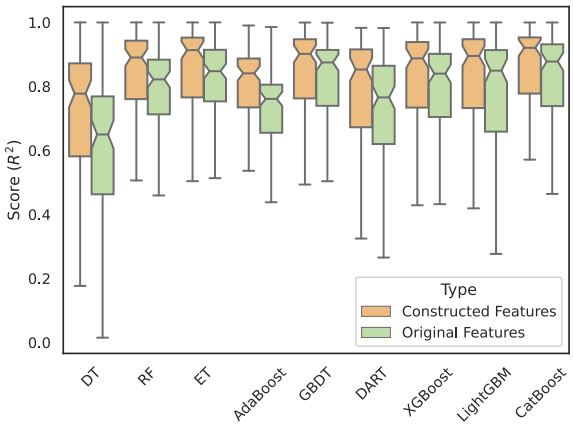


Fig. 16. R^2 performance with respect to different algorithms when using constructed features or original features.

VII. CONCLUSION

The aim of this article is to proposed an EF for regression problems using the GP method, which is capable of discovering the most expressive features in addressing regression problems for users without domain knowledge of specific tasks. We accomplish this goal by evolving several sets of discriminatory features using the multitree GP. For each set of discriminatory features synthesized by a multitree GP, a DT can be built based on these features. Based on that, the fitness vector of each GP individual can be calculated through the cross-validation scheme on all the training data points. Then, the promising GP individuals are chosen for the next generation by the lexicase selection operator and stored in an archive. Finally, we build DTs based on the GP individuals stored in the archive to form a forest, which we then use for further prediction. The proposed algorithm is tested against ten state-of-the-art machine learning algorithms, including four bagged DT methods and six boosted DT methods, using 117 benchmark datasets. The experimental results show that the EF outperforms all other machine learning algorithms in terms of generalization performance.

There are several research directions worthwhile to explore in the future. First, it should improve the performance of the EF on large datasets. Furthermore, how to apply the proposed method to classification problems should be investigated. Besides that, incorporating existing high-dimensional feature extraction techniques from image processing and bioinformatics domains into EF to extend this algorithm to broader

Authorized licensed use limited to: Universitas Indonesia. Downloaded on November 17,2023 at 02:05:24 UTC from IEEE Xplore. Restrictions apply.

domains is also a promising direction that needs to be explored.

REFERENCES

[1] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001.

[2] P. Geurts, D. Ernst, and L. Wehenkel, "Extremely randomized trees," *Mach. Learn.*, vol. 63, no. 1, pp. 3–42, 2006.

[3] T. Rainforth and F. D. Wood, "Canonical correlation forests," 2015, *arXiv:1507.05444*.

[4] H. Elghazel, A. Aussem, and F. Perraud, "Trading-off diversity and accuracy for optimal ensemble tree selection in random forests," in *Ensembles in Machine Learning Applications*. Heidelberg, Germany: Springer, 2011, pp. 169–179.

[5] A. Krogh and J. Vedelsby, "Neural network ensembles, cross validation, and active learning," in *Proc. Int. Conf. Adv. Neural Inf. Process. Syst. (NIPS)*, Denver, CO, USA, 1994, pp. 231–238.

[6] J. R. Quinlan *et al.*, "Learning with continuous classes," in *Proc. 5th Aust. Joint Conf. Artif. Intell.*, vol. 92. 1992, pp. 343–348.

[7] D. C. Wickramarachchi, B. L. Robertson, M. Reale, C. J. Price, and J. Brown, "HHCART: An oblique decision tree," *Comput. Stat. Data Anal.*, vol. 96, pp. 12–23, Apr. 2016.

[8] F. Avellaneda, "Efficient inference of optimal decision trees," in *Proc. AAAI Conf. Artif. Intell.*, vol. 34, 2020, pp. 3195–3202.

[9] G. Aglin, S. Nijssen, and P. Schaus, "Learning optimal decision trees using caching branch-and-bound search," in *Proc. AAAI Conf. Artif. Intell.*, vol. 34, 2020, pp. 3146–3153.

[10] A. Paez, F. López, M. Ruiz, and M. Camacho, "Inducing non-orthogonal and non-linear decision boundaries in decision trees via interactive basis functions," *Expert Syst. Appl.*, vol. 122, pp. 183–206, May 2019.

[11] R. Katuwal, P. N. Suganthan, and L. Zhang, "Heterogeneous oblique random forest," *Pattern Recognit.*, vol. 99, Mar. 2020, Art. no. 107078.

[12] K. Nag and N. R. Pal, "Feature extraction and selection for parsimonious classifiers with multiobjective genetic programming," *IEEE Trans. Evol. Comput.*, vol. 24, no. 3, pp. 454–466, Jun. 2020.

[13] Y. Bi, B. Xue, and M. Zhang, "Genetic programming with image-related operators and a flexible program structure for feature learning in image classification," *IEEE Trans. Evol. Comput.*, vol. 25, no. 1, pp. 87–101, Feb. 2021.

[14] B. Al-Helali, Q. Chen, B. Xue, and M. Zhang, "Multi-tree genetic programming for feature construction-based domain adaptation in symbolic regression with incomplete data," in *Proc. Genet. Evol. Comput. Conf.*, 2020, pp. 913–921.

[15] A. Lensen, B. Xue, and M. Zhang, "Genetic programming for evolving similarity functions for clustering: Representations and analysis," *Evol. Comput.*, vol. 28, no. 4, pp. 531–561, 2020.

[16] K. Neshatian, M. Zhang, and P. Andreae, "A filter approach to multiple feature construction for symbolic learning classifiers using genetic programming," *IEEE Trans. Evol. Comput.*, vol. 16, no. 5, pp. 645–661, Oct. 2012.

[17] B. Tran, B. Xue, and M. Zhang, "Genetic programming for feature construction and selection in classification on high-dimensional data," *Memetic Comput.*, vol. 8, no. 1, pp. 3–15, 2016.

[18] B. Tran, B. Xue, and M. Zhang, "Genetic programming for multiple-feature construction on high-dimensional classification," *Pattern Recognit.*, vol. 93, pp. 404–417, Sep. 2019.

[19] W. La Cava and J. H. Moore, "Learning feature spaces for regression with genetic programming," *Genet. Program. Evol. Mach.*, vol. 21, pp. 433–467, Mar. 2020.

[20] M. Virgolin, T. Alderliesten, and P. A. N. Bosman, "On explaining machine learning models by evolving crucial and compact features," *Swarm Evol. Comput.*, vol. 53, Mar. 2020, Art. no. 100640.

[21] B. H. Menze, B. M. Kelm, D. N. Splitthoff, U. Koethe, and F. A. Hamprecht, "On oblique random forests," in *Proc. Joint Eur. Conf. Mach. Learn. Knowl. Discov. Databases*, 2011, pp. 453–469.

[22] L. Zhang and P. N. Suganthan, "Random forests with ensemble of feature spaces," *Pattern Recognit.*, vol. 47, no. 10, pp. 3429–3437, 2014.

[23] L. Zhang and P. N. Suganthan, "Oblique decision tree ensemble via multisurface proximal support vector machine," *IEEE Trans. Cybern.*, vol. 45, no. 10, pp. 2165–2176, Oct. 2015.

[24] T. M. Tomita *et al.*, "Sparse projection oblique randomer forests," *J. Mach. Learn. Res.*, vol. 21, no. 104, pp. 1–39, 2020.

[25] J. J. Rodriguez, L. I. Kuncheva, and C. J. Alonso, "Rotation forest: A new classifier ensemble method," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 10, pp. 1619–1630, Oct. 2006.

[26] A. Bagnall, M. Flynn, J. Large, J. Line, A. Bostrom, and G. Cawley, "Is rotation forest the best classifier for problems with continuous features?" 2018, *arXiv:1809.06705*.

[27] M. Norouzi, M. Collins, M. A. Johnson, D. J. Fleet, and P. Kohli, "Efficient non-greedy optimization of decision trees," in *Advances in Neural Information Processing Systems*, vol. 28. Red Hook, NY, USA: Curran, 2015, pp. 1729–1737.

[28] G.-H. Lee and T. S. Jaakkola, "Oblique decision trees from derivatives of ReLU networks," in *Proc. Int. Conf. Learn. Represent.*, 2019.

[29] M. A. Carreira-Perpinán and P. Tavallali, "Alternating optimization of decision trees, with application to learning sparse oblique trees," in *Advances in Neural Information Processing Systems*, vol. 31. Red Hook, NY, USA: Curran, 2018, pp. 1211–1221.

[30] S. R. Bulò and P. Kontschieder, "Neural decision forests for semantic image labelling," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Columbus, OH, USA, 2014, pp. 81–88.

[31] H. Iba, "Bagging, boosting, and bloating in genetic programming," in *Proc. 1st Annu. Conf. Genet. Evol. Comput.*, 1999, pp. 1053–1060.

[32] G. Dick, C. A. Owen, and P. A. Whigham, "Evolving bagging ensembles using a spatially-structured niching method," in *Proc. Genet. Evol. Comput. Conf.*, 2018, pp. 418–425.

[33] N. M. Rodrigues, J. E. Batista, and S. Silva, "Ensemble genetic programming," in *Genetic Programming (EuroGP)* (Lecture Notes in Computer Science, 12101). Cham, Switzerland: Springer, 2020, pp. 151–166.

[34] Y. Bi, B. Xue, and M. Zhang, "Genetic programming with a new representation to automatically learn features and evolve ensembles for image classification," *IEEE Trans. Cybern.*, vol. 51, no. 4, pp. 1769–1783, Apr. 2020.

[35] Y. Bi, B. Xue, and M. Zhang, "A divide-and-conquer genetic programming algorithm with ensembles for image classification," *IEEE Trans. Evol. Comput.*, vol. 25, no. 6, pp. 1148–1162, Dec. 2021.

[36] M. Virgolin, "Genetic programming is naturally suited to evolve bagging ensembles," in *Proc. Genet. Evol. Comput. Conf.*, 2021, pp. 830–839.

[37] S. Wang, Y. Mei, and M. Zhang, "Novel ensemble genetic programming hyper-heuristics for uncertain capacitated arc routing problem," in *Proc. Genet. Evol. Comput. Conf.*, 2019, pp. 1093–1101.

[38] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discov. Data Min.*, 2016, pp. 785–794.

[39] M. Virgolin, T. Alderliesten, A. Bel, C. Witteveen, and P. A. N. Bosman, "Symbolic regression and feature construction with GP-GOMEA applied to radiotherapy dose reconstruction of childhood cancer survivors," in *Proc. Genet. Evol. Comput. Conf.*, 2018, pp. 1395–1402.

[40] B. Pes, "Ensemble feature selection for high-dimensional data: A stability analysis across multiple domains," *Neural Comput. Appl.*, vol. 32, no. 10, pp. 5951–5973, 2020.

[41] J. R. Koza and J. R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, vol. 1. Cambridge, MA, USA: MIT Press, 1992.

[42] J. R. Quinlan, *C4.5: Programs for Machine Learning*. San Mateo, CA, USA: Morgan Kaufmann, 1993.

[43] W. La Cava, L. Spector, and K. Danai, "Epsilon-Lexicase selection for regression," in *Proc. Genet. Evol. Comput. Conf.*, 2016, pp. 741–748.

[44] C. A. Owen, G. Dick, and P. A. Whigham, "Characterizing genetic programming error through extended bias and variance decomposition," *IEEE Trans. Evol. Comput.*, vol. 24, no. 6, pp. 1164–1176, Dec. 2020.

[45] R. S. Olson, W. La Cava, P. Orzechowski, R. J. Urbanowicz, and J. H. Moore, "PMLB: A large benchmark suite for machine learning evaluation and comparison," *BioData Min.*, vol. 10, no. 1, pp. 1–13, 2017.

[46] Z.-H. Zhou and J. Feng, "Deep forest," *Nat. Sci. Rev.*, vol. 6, no. 1, pp. 74–86, 2019.

[47] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *J. Comput. Syst. Sci.*, vol. 55, no. 1, pp. 119–139, 1997.

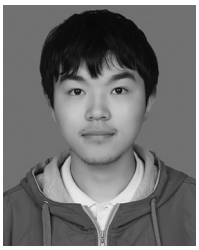
[48] R. K. Vinayak and R. Gilad-Bachrach, "DART: Dropouts meet multiple additive regression trees," in *Proc. Int. Conf. Artif. Intell. Stat.*, 2015, pp. 489–497.

[49] G. Ke *et al.*, "LightGBM: A highly efficient gradient boosting decision tree," in *Advances in Neural Information Processing Systems*, vol. 30. Red Hook, NY, USA: Curran, 2017, pp. 3146–3154.

[50] L. Prokhorenkova, G. Gusev, A. Vorobev, A. V. Dorogush, and A. Gulin, "CatBoost: Unbiased boosting with categorical features," in *Advances in Neural Information Processing Systems*. Red Hook, NY, USA: Curran, 2018, pp. 6638–6648.

Authorized licensed use limited to: Universitas Indonesia. Downloaded on November 17,2023 at 02:05:24 UTC from IEEE Xplore. Restrictions apply.

[51] J. Ni, R. H. Drieberg, and P. I. Rockett, "The use of an analytic quotient operator in genetic programming," *IEEE Trans. Evol. Comput.*, vol. 17, no. 1, pp. 146–152, Feb. 2013.
[52] M. Nicolau and A. Agapitos, "Choosing function sets with better generalisation performance for symbolic regression models," *Genet. Program. Evol. Mach.*, vol. 22, pp. 73–100, May 2020.
[53] W. La Cava *et al.*, "Contemporary symbolic regression methods and their relative performance," 2021, *arXiv:2107.14351*.
[54] N. Li, Y. Yu, and Z.-H. Zhou, "Diversity regularized ensemble pruning," in *Proc. Eur. Conf. Mach. Learn. Knowl. Discov. Databases*, 2012, pp. 330–345.
[55] A. Rahimi and B. Recht, "Weighted sums of random kitchen sinks: Replacing minimization with randomization in learning," in *Advances in Neural Information Processing Systems (NIPS)*. Red Hook, NY, USA: Curran Assoc., Inc., 2008, pp. 1313–1320.
[56] W. La Cava and J. H. Moore, "Ensemble representation learning: An analysis of fitness and survival for wrapper-based genetic programming methods," in *Proc. Genet. Evol. Comput. Conf.*, 2017, pp. 961–968.



Hengzhe Zhang received the B.Sc. degree in software engineering from Xiangtan University, Xiangtan, China, in 2019. He is currently pursuing the master's degree with East China Normal University, Shanghai, China.

His current research interests include symbolic regression, genetic programming, evolution computation, and statistical machine learning.

Aimin Zhou (Senior Member, IEEE) received the B.Sc. and M.Sc. degrees in computer science from Wuhan University, Wuhan, China, in 2001 and 2003, respectively, and the Ph.D. degree in computer science from the University of Essex, Colchester, U.K., in 2009.

He is currently a Professor with the School of Computer Science and Technology, Shanghai Institute of AI for Education, East China Normal University, Shanghai, China. He has authored over 80 peer-reviewed papers. His current research interests include evolutionary computation and optimization, machine learning, image processing, and their applications.

Prof. Zhou is an Associate Editor of *Swarm and Evolutionary Computation* and an Editorial Board Member of *Complex and Intelligent Systems*.

Hu Zhang received the M.Sc. degree in mechanical design and theory from China Three Gorges University, Yichang, China, in 2012, and the Ph.D. degree from the Center for Control Theory and Guidance Technology, Harbin Institute of Technology, Harbin, China, in 2016.

He is currently a Senior Engineer with Beijing Electro-Mechanical Engineering Institute, Beijing, China. His current research interests include evolutionary computation, military intelligence, optimal design of system, and operation assessment.

Authorized licensed use limited to: Universitas Indonesia. Downloaded on November 17,2023 at 02:05:24 UTC from IEEE Xplore. Restrictions apply.