JOURNAL OF INFORMATION AND COMMUNICATION TECHNOLOGY
http://e-journal.uum.edu.my/index.php/jict

How to cite this article:
Kumar, S., Ratnoo, S., & Vashishtha, J. (2021). Hyper heuristic evolutionary approach for constructing decision tree classifiers. *Journal of Information and Communication Technology, 20*(2), 249-276. https://doi.org/10.32890/jict2021.20.2.5

Hyper-Heuristic Evolutionary Approach for Constructing Decision Tree Classifiers

Sunil Kumar, Saroj Ratnoo & Jyoti Vashishtha
Department of Computer Science and Engineering
Guru Jambheshwar University of Science &
Technology, India

skvermacse@gmail.com
ratnoosaroj; jyoti.vst@gmail.com

Received: 7/7/2020 Revised: 28/10/2020 Accepted: 1/11/2020 Published: 23/2/2021

ABSTRACT

Decision tree models have earned a special status in predictive modeling since these are considered comprehensible for human analysis and insight. Classification and regression tree (CART) algorithm is one of the renowned decision tree induction algorithms to address classification as well as regression problems. Finding optimal values for the hyper parameters of a decision tree construction algorithm is a challenging issue. While making an effective decision tree classifier with high accuracy and comprehensibility, there is a need to address the question of setting optimal values for its hyper parameters like the maximum size of the tree, the minimum number of instances required in a node for inducing a split, node splitting criterion, and the amount of pruning. The hyper parameter setting influences the performance of the decision tree model. As known by researchers, there is no single setting of hyper parameters that works equally well for different datasets. A particular setting that gives an optimal decision tree for one dataset may produce a sub-optimal decision tree model for another dataset. In this paper, a hyper-heuristic approach was presented for tuning the hyper parameters of recursive and partition trees (Rpart), which is a typical implementation of CART in statistical and data analytics package R. The study employed an evolutionary algorithm as hyper-heuristic for tuning the hyper parameters of the decision tree classifier. The approach was named as hyper-heuristic evolutionary approach with recursive and partition trees (HEARpart). The proposed approach was validated on 30 datasets. It was statistically proven that HEARpart performed significantly better than WEKA's J48 algorithm in terms of error rate, F-measure, and tree size. Furthermore, the suggested hyper-heuristic algorithm constructed significantly comprehensible models as compared to WEKA's J48, CART, and other similar decision tree construction strategies. The results showed that the accuracy achieved by the hyper-heuristic approach was slightly less as compared to the other comparative approaches.

Keywords: Machine learning, evolutionary algorithms, hyper-heuristic, decision trees, classification, CART.

INTRODUCTION

Decision trees, a class of supervised learning algorithms, are widely used for addressing classification problems in data mining and machine learning. A decision tree classifier is built from the training data of class-labeled instances. Subsequently, the classifier is used to predict the class labels for new instances. Decision trees have come to be one of the most widely used predictive modeling algorithms due to their reasonable accuracy and high comprehensibility. Since Quinlan first proposed the decision tree induction algorithm (Quinlan, 1986), many variants of decision trees such as C4.5, C5.0, chi-square automatic interaction detection (CHAID), and classification and regression tree (CART), algorithm CART have appeared for addressing predictive modeling tasks (Han et al., 2011; Witten & Frank, 2011).

The classical methods create decision tree models by following a top-down greedy induction approach. Initially, the root node contains all the data. The data at the root node is partitioned into sub-nodes based on the values of an attribute selected for the split. The same process repeats recursively for each sub-node in a hierarchical manner until a stopping criterion is met. There are many criteria in the literature for the selection of the splitting attributes. Some of the popular ones are entropy, information gain, and Gini index (Han et al., 2011; Witten & Frank, 2011).

A decision tree is constructed in two phases: i) a learning phase and ii) a pruning phase. The learning phase constructs the tree in a top-down recursive manner as described in the previous paragraph. Since the initial decision tree could be large and complex, the pruning phase applies heuristics to reduce the complexity of the tree. Therefore, all decision tree algorithms adopt pruning to avoid overfitting of the model (Esposito et al., 1997). Determining the amount of pruning is a complex issue. If a tree is not pruned enough, it tends to capture the noise in the training data. Such an overfitted model has less generalization power and higher variance. Over-pruning may result in a model with higher bias and poor accuracy.

Breiman et al. (1984) introduced classification and regression tree, which is abbreviated as CART. The decision tree algorithms C4.5 and C5.0 use entropy as a measure of impurity while splitting nodes, whereas CART employs the Gini index, which is a generalization of binomial variance. Therefore, CART can have only binary splits as compared to other decision tree algorithms that allow multiway splits. While most of the decision tree algorithms are meant for classification problems, CART can be used for classification and regression modeling with equal ease and efficiency. The CART algorithm is implemented with slight variations using R package, named as recursive and partition tree (*Rpart*). One of the variations is that *Rpart* is allowed to have either information gain or Gini index as the splitting criterion (Therneau et al., 2019).

The hyper parameters such as the size of neighborhood for k-nearest neighbor classifiers, choice of kernel function for support vector machines, and splitting criterion for decision trees cannot be determined from the training data. These have to be determined externally by the machine learning practitioners or researchers who tune these parameters experimentally. Almost all machine learning algorithms have one or more hyper parameters that directly influence their generalization power and predictive performance. Determining the appropriate values for hyper parameters is a challenging issue. Any ad hoc approach for setting these parameters results in the sub-optimal performance of the learning algorithms. No single hyper parameter setting works equally well across different datasets. A particular hyper parameter setting may produce an accurate classifier for a specific dataset, but it may fail in doing so for other datasets.

Like other machine learning algorithms, for optimal performance of decision tree classifiers, there is a need to set its several hyper parameters, such as the maximum depth of the tree, the minimum number of instances at a node for inducing a split, the splitting criterion, and the complexity parameter that controls the amount of pruning. From the same training data, many predictive models could be constructed depending on the choice of values of different hyper parameters. Obtaining a decision tree model with the highest possible accuracy and simplicity is not at all a trivial task and, in fact, finding an optimal decision tree model for classification is a complex combinatorial optimization problem. The decision tree model with high accuracy and comprehensibility cannot be obtained without addressing the question of setting optimal values for its various hyper parameters.

Evolutionary algorithms (EAs) work with a population of candidate solutions to optimization problems. Over subsequent generations, better solutions are evolved through fitness proportionate selection, recombination, and mutation operators (Michalewicz, 1996; Yu & Gen, 2010). Besides, these are effective hyper-heuristic methods for automatically designing decision tree models (Barros et al., 2012).

This paper proposes a hyper-heuristic evolutionary approach for constructing accurate and comprehensible decision tree models using the recursive and partition tree (*Rpart)* algorithm for a variety of datasets. The proposed algorithm is named as hyper-heuristic evolutionary approach with recursive and partition trees (*HEARpart*). A genetic algorithm (GA) is used as a hyper-heuristic to find optimal decision tree models for each dataset from the search space of all possible decision trees. The suggested approach tunes the four hyper parameters of recursive partitioning and regression tree: i) the minimum number of instances that must exist in a node for a split; ii) maximum depth of the tree; iii) splitting criteria; and iv) complexity parameter to control the amount of pruning.

The motivation of this experimental research is to improve accuracy and reduce the size of the decision trees classifier. This paper compares the decision tree models produced by the proposed hyper-heuristic evolutionary approach with the ones generated by *Rpart* with its hyper parameters set to their default values and with the other similar research works. The tuning of hyper parameters for each data domain exclusively during the learning phase of decision tree models has been successful in achieving statistically higher comprehensibility without a significant compromise on the predictive performance. The main contribution of the suggested approach is toward building decision tree models that are optimal for each dataset individually in terms of accuracy and comprehensibility. While *Rpart* is used for construction of decision trees, the approach can easily be migrated to other machine learning algorithms.

The rest of the paper is organized as follows. The second section presents a review of the earlier research works carried out for constructing decision tree predictive models using meta or hyper-heuristic approaches. The third section describes the components of the hyper-heuristic approach proposed in this paper. The experimental design consisting of the research methodology is given in the fourth section. A discussion and analysis of results are presented in the fifth section. The last section concludes the research and points toward the novel research directions.

RELATED WORKS

This section reviews the related research works to contextualize the present work. The section intends to include only the benchmark and the most cited works. The review focuses on the application of metaheuristic and hyper-heuristic approaches for decision tree construction. In the world of machine learning and predictive analytics, decision trees are one of the most popular prediction methods. Many decision tree algorithms such as ID3, C4.5, C5.0, CART, and CHAID have been developed (Han et al., 2011; Witten & Frank, 2011). The decision tree models follow a greedy strategy for growing from top to bottom, which often results in local optimal predictive models.

Researchers have proposed various methods to address the problems posed by the top-down recursive greedy strategy of decision tree construction. Two of these methods are ensemble learning and the use of metaheuristics and hyper-heuristics for constructing optimal decision tree models. In ensemble learning, many trees are grown, and the outcome of the model is predicted through voting amongst all the trees included in the ensemble. The accuracy of ensemble methods is higher than the models based on a single decision tree. However, in ensemble learning, the comprehensibility of a single decision tree model is compromised. Therefore, ensembles are not a good choice for domains such as disease and fault diagnosis where the interpretability of classifier is critical for decision-making (Cutler et al., 2012; Polikar, 2012). The aim of metaheuristics and hyper-heuristics approaches is to maintain the high accuracy as well as comprehensibility of decision tree models. The EAs fall under the category of metaheuristic algorithms and perform a comprehensive search in the search space of all the possible decision tree classifiers to arrive at the global optimal solution. The EAs also tackle the problem of attribute interaction, whereas greedy methods of decision tree construction fail to do so (Barros et al., 2012).

The EAs have been used mainly in three ways for the construction of decision tree models. In the first approach, researchers have tried to construct an optimal decision tree classifier by combining the portions of the existing decision trees by using evolutionary operators. In the second approach, metaheuristics have been applied for building decision tree models from scratch. In the third approach, meta or hyper-heuristic algorithms have been applied to automatically design decision tree algorithms by optimizing the hyper parameters such as node splitting criterion, complexity of decision tree, selection of features, selection of appropriate training and test data, or a combination of these.

Fue et al. (2003) followed the first approach and applied a GA for obtaining an accurate decision tree classifier. In their work, the initial population of trees was generated using C4.5. Later, the components of various trees were combined to obtain better decision tree classifiers using GA operators. There have been attempts to arrange decision rules in the hierarchical form to reduce redundancy and increase comprehensibility (Bharadwaj & Saroj, 2009; 2010). The authors applied GAs to discover classification rules at multiple levels of abstraction. Liu and Fan (2014) employed a GA to optimize the classification rules obtained from the C4.5 decision tree algorithm for classifying mobile users. The research was shown to achieve higher accuracy and comprehensibility as compared to the simple decision tree and support vector machine classifiers. All the above approaches optimize the existing classifiers. However, these approaches only combine the portions of existing decision trees or classification rules to arrive at classifiers with improved performance. Such approaches fail to find the optimal classifier in case the essential components to form it are missing in the existing population of decision trees.

In the second approach, the decision trees are constructed right from scratch using metaheuristic approaches. In this direction, Cha and Tappert (2009) devised encoding and decoding schemes for decision trees to be used with metaheuristic methods. Pacheco et al. (2012) suggested a greedy randomized adaptive search procedure (GRASP) for constructing binary classification trees to produce accurate yet simple decision tree models. The authors modified the search procedure for determining the best splitting attribute at each node of the decision tree construction process. Instead of selecting the single best splitting attribute based on some impurity reduction measure, GRASP randomly selected one of the best splitting attributes from a set of feasible ones. The process grew several trees and the decision tree with the least complexity was selected. The focus of this work was on discovering the most comprehensible decision tree classifiers for a predetermined level of accuracy. Hemmateenejad et al. (2011) combined the ant colony optimization algorithm and crossover and mutation operators from GA to improve the performance of CART predictive models. The hybrid approach was shown to have achieved better predictive performance for modeling the melting points of a large number of chemical compounds.

Further, in the sequence, a significant contribution came from Otero et al. (2012) for constructing decision trees by combining the strategy of classical tree induction techniques with the ant colony optimization (ACO) algorithm. The authors did extensive experimentation with the help of 22 publicly available datasets from the UCI machine learning repository. Their work showed that the predictive accuracy achieved by their approach was significantly higher than the accuracy of C4.5 and CART. Decision tree classifiers based on GAs have achieved competitive accuracy for network intrusion detection and fault diagnosis (Karabadji et al., 2014, 2012; Stein et al., 2005). Recently, Adibi (2019) suggested a GA-based optimal decision tree construction method to improve the performance of decision tree classifiers for single and multi-output datasets. The authors presented a bi-level discrete-continuous GA to simultaneously select effective features and construct an optimal tree.

The main advantage of evolving decision tree classifiers from scratch by using metaheuristic approaches such as EA and ACO is to escape from the local optimal solutions. The literature review shows that the metaheuristic approaches are less prone to sub-optimal solutions and can find optimal decision tree classifiers with high accuracy and comprehensibility. Nevertheless, the metaheuristic approaches are not without limitations. Domain-specific and particular heuristic and metaheuristic methods do not often perform well when applied to diverse problem domains without significant modification. Therefore, hyper-heuristics approaches have recently gained increased attention of machine learning researchers as a third approach for constructing optimal decision tree classifiers.

Hyper-heuristics are search methods that operate on a lower level of heuristics and have emerged as a way to enhance the generalization capabilities of machine learning algorithms. In this direction, an outstanding contribution came about in the form of a hyper-heuristic evolutionary algorithm (HEAD-DT) for the automatic designing of decision tree algorithms (Barros et al., 2012; Barros et al., 2014). The automatically designed decision tree algorithms were devised by combining building blocks of heuristics through an evolutionary algorithm. The building blocks included splitting genes, stopping criteria genes, missing value genes, and pruning genes. This hyper-heuristic approach surpassed the traditional decision tree construction methods like C4.5 and CART on account of accuracy and F-measure. Mantovani et al. (2016) explored different hyper parameter tuning techniques for WEKA's J48 decision tree algorithm. The authors used grid search, GA, particle swarm optimization, and estimation of distribution algorithm as the three metaheuristic techniques for tuning of hyper parameters of J48 algorithm. The results showed that all the tuning techniques were at par with each other, but performed significantly better as compared to the decision tree models generated by simply using the default hyper parameters of J48 algorithm. Karabadji et al. (2017) proposed an evolutionary scheme (ES) for identifying the best training and test sets, and other parameters to pull out the optimal decision tree. The authors showed that their approach outperformed the classical decision tree construction methods in terms of accuracy and simplicity. Since hyper-heuristic methodologies provided domain-independent solutions, the researchers continued to develop hyper-heuristic methods that applied to a range of different problems (Drake et al., 2020; Yafrani et al., 2018; Sabar et al., 2017).

The experimental research conducted in this paper is an attempt to improve the accuracy and comprehensibility of the *Rpart* decision tree classifier by tuning its hyper parameters using a GA that is a promising evolutionary optimization technique. This study intends to discover the optimal decision tree classifiers that are well fitted to the datasets across different domains. The research presented in this paper falls within the third approach of automatically discovering optimal decision tree classification models.

THE PROPOSED HYPER-HEURISTIC APPROACH

This section presents the overall design of the proposed hyper-heuristic approach (*HEARpart*). To begin with, the dataset was divided into three parts: i) training dataset, ii) validation dataset, and iii) test dataset, by using uniform random sampling without replacement. The data partitions are shown in Figure 1. The hyper parameters of the *Rpart* construction algorithm were tuned on the validation dataset by using GA, specifically designed for this purpose. The GA routine returned the set of optimal values for hyper parameters. Thereupon a decision tree model was constructed on the training data by setting the hyper parameters of *Rpart* to an optimal configuration. The performance of the resulting model was evaluated on the test data. The overall design of the proposed hyper-heuristic method is illustrated with the help of a flow chart in Figure 2.

Figure 1

The Data Partitioning Scheme.

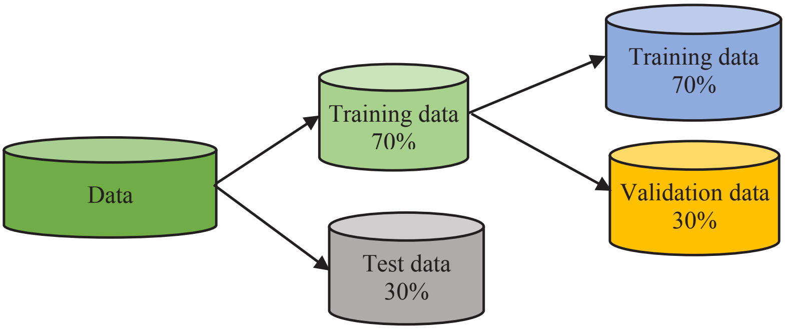
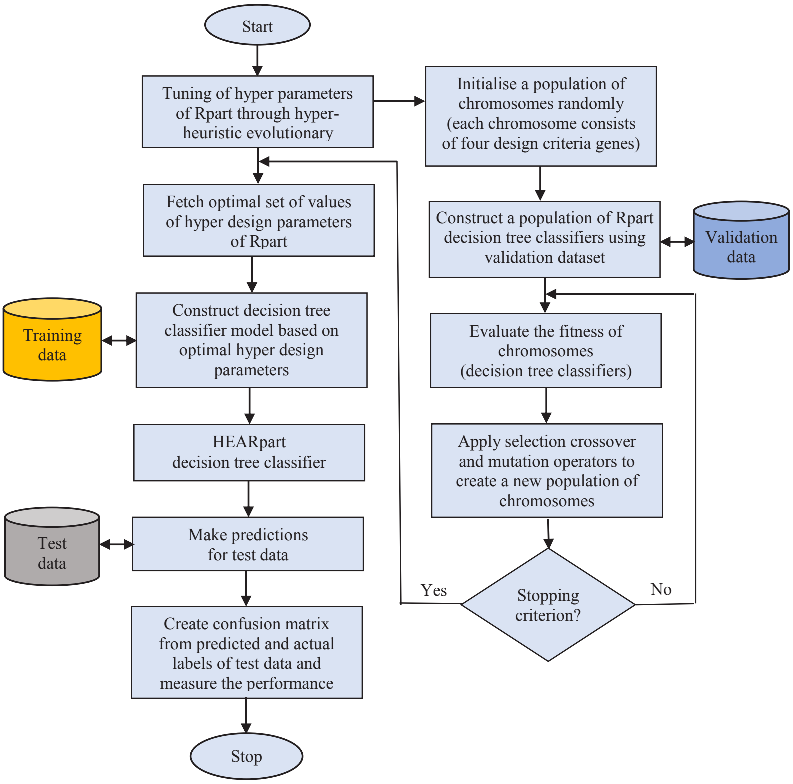


Figure 2

The Proposed System (HEARpart).


Start
Tuning of hyper parameters of Rpart through hyper-heuristic evolutionary
Initialise a population of chromosomes randomly (each chromosome consists of four design criteria genes)
Fetch optimal set of values of hyper design parameters of Rpart
Construct a population of Rpart decision tree classifiers using validation dataset
Validation data
Training data
Construct decision tree classifier model based on optimal hyper design parameters
Evaluate the fitness of chromosomes (decision tree classifiers)
HEARpart decision tree classifier
Apply selection crossover and mutation operators to create a new population of chromosomes
Test data
Make predictions for test data
Yes
Stopping criterion?
No
Create confusion matrix from predicted and actual labels of test data and measure the performance
Stop

The tuning of the hyper parameter of a decision tree algorithm was at the core of the *HEARpart*. For the hyper-heuristic approach, a GA was designed to tune the hyper parameters of the *Rpart* for a variety of datasets. This study chose the following four hyper parameters that directly influence the performance of decision tree models.

Minimum split: It signifies the minimum number of instances that must exist in a node for further splitting. The optimal value of this parameter will depend on the number of instances in the dataset and the distribution of the target variable.

Complexity: The role of this factor is to prune off the worthless splits to enhance the overall fit of the model. The factor plays an important role in avoiding the splits that tend to produce overfitted models and thus reduce the generalization power of the model. However, an over-pruned tree will not fit the training data and will be less accurate. The setting up of an appropriate value for the complexity of the model is of utmost importance.

Depth of the tree: This puts a maximum limit on the levels of the decision tree from the root node to leaf nodes. The depth of a tree is an indicator of the comprehensibility of the decision tree classifiers. If two decision trees have the same accuracy, the smaller tree is more desirable.

Splitting criterion: The choice of the splitting criterion is another important issue in the construction of decision tree models. This may vary from one dataset to another. The *Rpart* algorithm provides information gain and Gini index as the two splitting criteria. The GA can use either of the two.

The Genetic Algorithm Designs

This section elaborates on the GA design for finding the optimal values for hyper parameters of the *Rpart* decision tree learning algorithm.

Encoding

Each chromosome in the GA population consisted of four genes (minimum split, complexity, maximum depth of the decision tree, and node splitting criterion), which have been explained earlier. This research encoded each chromosome as a numeric string of four genes, each one corresponding to the four criteria for decision tree construction. Each gene was assigned values out of the supported range of values for the respective criteria. The ranges of supported values of the genes were ‘1 to 100’, ‘0.01 to 0.3’, ‘2 to 20’, and ‘0 or 1’, respectively. A typical chromosome is shown in the form of a table.

The chromosome shown in Table 1 created an example decision tree according to the above-specified values of the various genes. This decision tree needed to have at least twenty examples for further splitting at any node and its height could not exceed the seventh level when the root of the tree was considered to be at level 0. The splitting criterion was 1 for information gain and 0 for the Gini index. The value of complexity parameter 0.10 means that all those splits would be pruned off, which would not enhance the fit of the model by 10%.

Table 1

Chromosome Structure

Minimum split	Complexity parameter	Maximum height of tree	Splitting criteria
20	0.10	7	1

Population Initialization

The population of the evolutionary algorithm was randomly initialized with the values from the supported ranges of the respective genes.

Fitness Function

The choice of the fitness function was very important as it directed the search toward the optimal solution. The GA used the accuracy of the tree as fitness. The accuracy of a classifier model can be defined as the percentage of correctly classified instances. The formula to calculate accuracy is given in Equation 1 below:

$$Accuracy = \frac{(TP + TN)}{(TP + TN + FP + FN)} \tag{1}$$

where TP, FN, FP, and TN represent the number of True Positives, False Negatives, False Positives, and True Negatives, respectively. These are defined below:

TP: These are the number of instances that actually belong to the positive class and also get predicted as positive by the classifier.

FN: These are the number of instances that actually belong to the positive class but get predicted as negative by the classifier.

FP: These are the number of instances that belong to the negative class but get predicted as positive by the classifier.

TN: These are the number of instances belong to the negative class and also get predicted negative by the classifier.

Genetic Operators

The genetic operators were used to modify or re-combine the genetic material of the GA population and introduce new genetic material. The present study employed the following genetic operators.

Selection

The suggested GA design applied the tournament selection strategy for creating the mating pool for the next generation. The tournament selection method avoided local convergence by keeping the selection pressure in control. In a tournament selection, 'K' individuals were randomly selected from the population and the individual with the best fitness was selected to become a parent. The same process was repeated to select another parent. The value of K was set to 4.

Crossover

In the proposed GA design, a single point heuristic crossover operator was applied for creating offspring from the real coded chromosomes. Out of the two parents P and Q, and assuming that Q was the better fit parent, the heuristic crossover created offspring as given in Equations 2 and 3.

$$O_1 = \alpha(Q - P) + Q \tag{2}$$

$$O_2 = \alpha Q + (1 - \alpha)P \tag{3}$$

The symbol in the above equations is a random number between 0 and 1. The first offspring is a projection, whereas the second one is a convex combination of the values of the two parents. For this reason, the heuristic crossover operator maintained diversity in the population as well as directed the search toward favorable zones of a search to arrive at the optimal solution. The genes after the crossover point were first modified according to Equations 2 and 3 and then these were swapped between the two parents.

Mutation

This research applied a random mutation, i.e., the gene values of chromosomes were randomly mutated within their respective ranges. The mutation operator was essential for exploring the search space of hyper parameters. The operator changed a small proportion of genes of the chromosomes to maintain genetic diversity in the GA population. Every gene in the chromosomes had a small probability to mutate.

Terminating Criterion

Genetic algorithms terminate either on the completion of a maximum number of iterations or when the diversity of its population reaches a pre-determined minimum. The GA was set to run for 100 generations; however, it terminated earlier if there was no improvement in fitness during the last 30 generations.

The real encoding of the chromosomes, the fitness function, and the respective GA operators described above created successive generations of decision tree classifiers based on different combinations of hyper parameter values. All the components of the GA worked in synergy and finally found an optimal set of hyper parameter values that created an optimum decision tree classifier in terms of accuracy and comprehensibility.

EXPERIMENTAL DESIGN AND RESULTS

This section describes the research methodology and presents the experimental results. It also includes the interpretation of the results.

Datasets

The performance of decision tree classifiers generated by *HEARpart* was tested on 30 datasets. The datasets were obtained from the UCI and Kaggle machine learning repositories. These datasets are shown in Table 2. The datasets were diverse in terms of the number of instances, the number of attributes, and the number of classes.

Table 2

Description of Datasets for Constructing Decision Tree Classifiers

No.	Dataset	#Insts	#Attribs	# Classes	No.	Dataset	#Insts	#Attribs	#Classes
1	Abalone	4177	8	28	16	Glass	214	10	6
2	Audiology	226	69	23	17	Hepatitis	80	20	2

(continued)

No.	Dataset	#Insts	#Attribs	# Classes	No.	Dataset	#Insts	#Attribs	#Classes
3	Bank marketing	4521	17	2	18	Mushroom	5644	23	2
4	Banknote authentication	1372	5	2	19	Nursery	12960	9	5
5	Breast cancer Wisconsin	699	11	2	20	Parkinson's	195	23	2
6	BEPS	1525	10	2	21	Primary tumor	120	18	18
7	Breast cancer	286	10	2	22	Segment	2310	20	7
8	Bridge version1	107	12	6	23	Sick	3103	28	2
9	Bupa liver	345	7	2	24	Sonar	208	61	2
10	Car	1729	7	4	25	Vote	232	17	2
11	Credit data	4455	14	2	26	Waveform 5000	5000	41	3
12	Cylinder Bands	541	33	2	27	Wdbc-mod2	569	31	2
13	Dermatology	366	35	6	28	Wine	178	14	3
14	Diabetes	769	9	2	29	Wine-white	4898	12	7
15	E. coli	337	8	8	30	Wine-red	1599	12	6

Tools

Classification tree models were constructed in R using rpart, ga, and caret packages from the CRAN website (https://cran.r-project.org). The '*ga*' package in R provided many general-purpose functions for optimization using GAs. It includes various flexible sets of tools for solving continuous and discrete optimization problems. It offers several kinds of encoding schemes, a large set of operators for selection, recombination, and mutation steps for designing a GA. A researcher only needs to simply embed the objective function for the optimization problem at hand. The caret package was used for measuring performance evaluation of the classifiers. It has a function *confusionMatrix()* that takes the predicted and actual class labels and outputs confusion matrix along with several other evaluation metrics. This study also used WEKA for running the J48 decision tree classifier algorithm.

Performance Metrics

For comparison of the decision tree classifiers, this study used error rate (1-accuracy), F-measure, and decision tree size. The total number of nodes in it measured the size of a decision tree. The classification accuracy of a classifier could be biased toward the majority class. Therefore, it could be a misleading measure of the performance of a classification algorithm, particularly for datasets with class skew, i.e., datasets with a small number of examples for the minority class as compared to the majority class. This might be the case for the datasets related to disease or fault diagnosis. Therefore, this research also included F-measure for evaluating the performance of algorithms.

F-measure is the harmonic mean of precision and recall of a classifier. Precision considers the percentage of instances that belong to the positive class out of the total instances predicted as that of the positive class. Recall, also known as sensitivity, measures the percentage of instances that are predicted into the positive class out of the total number of instances belonging to the positive class in the test data. It is measured using Equations 4, 5, and 6.

$$F - Measure = \frac{2\ (Precision \times Sensitivity)}{Precision + Sensitivity} \tag{4}$$

where,

$$\mathrm{Pr}ecision = \frac{TP}{TP + FP} \tag{5}$$

$$Sensitivity = \frac{TP}{TP + FN} \tag{6}$$

The tree size indicated the comprehensibility of the decision tree classifiers. The lesser the size of the tree, the better the comprehensibility of the classifier.

Sampling Scheme

The present study adopted a holdout random method as the sampling scheme. The data were divided into training and test datasets in a ratio of 0.7:0.3. The training data were further divided into training-training and training-validation sets again in the ratio of 0.7: 0.3. The proposed classifier algorithm used the training-training data for its learning phase. The training-validation dataset was applied for tuning the various hyper parameters of the decision tree algorithm, while the test dataset was used for evaluating the resulting classifiers. Further, to balance out the influence of stochastic sampling error on results, the *HEARpart* was run ten times with different random seeds. The results reported in this research were based on the average values over the ten runs of the proposed algorithm. For *Rpart* and J48 decision tree classifiers, no validation dataset was required. Therefore, the datasets were divided simply into 67 percent and 33 percent as training and test data for these two classification algorithms.

Parameter Setting

The parameter setting for the GA is given in Table 3. A stopping criterion was also included such that if there was no improvement in the solution for the last 30 generations, the GA automatically stopped executing. The *Rpart* algorithm was run with its default values except for the four hyper parameters of the decision tree construction process that were optimized by *HEARpart*.

Table 3

Parameter Setting for HEARpart

Population size	Number of generations	Crossover rate	Elitism	Mutation rate
50	100	0.8	1	0.1

The parameter setting for J48 classification algorithm is given in Table 4. The other parameters were kept at their default values. The minimum number of objects required for further splitting a node was 20, which was also the default value for *Rpart* classifier in R. Since *HEARpart* always produced binary splits while constructing decision tree classifiers, the value for the binary split parameter of J48 was set to true to make a fair comparison for the size of decision tree classifiers.

Table 4

Parameters Setting for WEKA J48

Binary split	Confidence factor	Min num objects	Num fold	Batch size
True	0.2	20	2	32

Comparative Study

This research compared the decision tree classifiers generated by *HEARpart* with three benchmark research works in the related field. The proposed approach was compared to the hyper-heuristic evolutionary algorithm for the automatic designing of decision tree algorithm (HEAD-DT) (Barros et al., 2012). The suggested algorithm was also compared to the research work by Otero et al. (2012). In this paper, the authors used the ACO algorithm for decision tree induction in a top-down manner by probabilistically selecting attributes for node splits based on the amount of pheromone and heuristic information. A comparison was also made to the research work by Karabadji et al. (2017), who employed a novel evolutionary strategy for identifying the best parameter settings to construct optimal decision trees.

RESULTS AND ANALYSIS

Table 5 shows the comparative performance of *HEARpart* with the simple *Rpart* algorithm and J48 of WEKA on 30 datasets. The comparison of various decision tree classifiers was reported in terms of error rate, F-measure, and size of decision tree classifiers. The best values were highlighted in bold. Out of 30 the datasets, *HEARpart* had less or at par error rate on 22 datasets as compared to other algorithms. The WEKA-J48 results indicated less error rate for five datasets and *Rpart* gave less error rate only on three datasets. Concerning F-measure, *HEARpart* performed better or at par with the other algorithms again in 22 datasets, whereas WEKA-J48 only won in 3 datasets. The results of *HEARpart* appeared significant on comprehensibility as well.

Table 5

The Performance Evaluation of the Three Algorithms

Datasets	Classification error			F-measure			Decision tree		
	J48	Rpart	HEARpart	J48	Rpart	HEARpart	J48	Rpart	HEARpart
Abalone	0.75 (3)	**0.76** (1.5)	**0.76** (1.5)	0.23 (3)	0.33 (2)	**0.34** (1)	169 (3)	9(2)	**8**(1)
Audiology	**0.48 (1)**	**0.48** (1)	**0.48** (1)	0.43 (3)	0.61 (2)	**0.68** (1)	7(1)	**9**(2.5)	**9**(2.5)
Bank marketing	**0.10 (1.5)**	0.09 (3)	**0.10** (1.5)	0.88 (3)	**0.92** (1.5)	**0.92** (1.5)	15(2)	23(3)	**13**(1)
Banknote authentication	**0.02** (1)	0.04 (2)	0.05 (3)	**0.98** (1.5)	**0.98** (1.5)	0.96 (3)	19(2.5)	19(2.5)	**13**(1)
Breast cancer Wisconsin	0.05 (3)	0.04 (2)	**0.03** (1)	0.95 (3)	0.96 (2)	**0.97** (1)	**5**(1)	8(3)	6(2)
BEPS	0.45 (3)	0.43 (2)	**0.42** (1)	0.54 (3)	0.76 (2)	**0.77** (1)	27(3)	17(2)	**5**(1)
Breast cancer	**0.27** (1.5)	0.36 (3)	**0.27** (1.5)	0.69 (3)	0.94 (2)	**0.95** (1)	**5**(1.5)	12(3)	**5**(1.5)
Bridge version1	0.76 (3)	0.56 (2)	**0.46** (1)	0.20 (3)	0.55 (2)	**0.64** (1)	**3**(1)	5(2)	7(3)
Bupa liver	0.36 (3)	**0.30** (1)	0.31 (2)	0.65 (2.5)	0.65 (2.5)	**0.66** (1)	15(2)	23(3)	**9**(1)
Car	0.11 (3)	**0.07** (1.5)	**0.07** (1.5)	0.89 (3)	**0.92** (1.5)	**0.92** (1.5)	37(3)	32(2)	**29**(1)
Credit data	0.20 (2.5)	0.20 (2.5)	**0.18** (1)	0.79 (3)	0.80 (2)	**0.82** (1)	15(2)	25(3)	**9**(1)
Cylinder bands	0.44 (3)	**0.33** (1.5)	**0.33** (1.5)	0.53 (3)	0.67 (2)	**0.70** (1)	11(2)	14(3)	**8**(1)

(continued

Datasets	Classification error			F-measure			Decision tree		
	J48	Rpart	HEARpart	J48	Rpart	HEARpart	J48	Rpart	HEARpart
Diabetes	0.27 (2.5)	0.27 (2.5)	**0.24** (1)	**0.73** (1.5)	**0.73** (1.5)	0.72 (3)	19(2)	24(3)	**13**(1)
E. coli	0.30 (3)	0.22 (2)	**0.21** (1)	0.66 (3)	0.79 (2)	**0.81** (1)	11(2.5)	11(2.5)	**10**(1)
Glass	0.48 (3)	**0.37** (1)	0.40 (2)	0.49 (3)	0.64 (2)	**0.65** (1)	7(1)	13(3)	11(2)
Hepatitis	0.25 (3)	0.23 (2)	**0.19** (1)	0.64 (3)	0.86 (2)	**0.90** (1)	**1**(1)	2(2)	3(3)
Mushroom	**0.00** (1)	**0.00** (1)	**0.00** (1)	**1.00** (1)	**1.00** (1)	**1.00** (1)	11(3)	**5**(1.5)	**5**(1.5)
Nursery	**0.04** (1)	0.13 (2.5)	0.13 (2.5)	**0.96** (3)	0.88 (1.5)	0.88 (1.5)	125(3)	**11**(1.5)	**11**(1.5)
Parkinson's	0.22 (3)	**0.11** (1)	0.14 (2)	0.78 (3)	**0.89** (1)	0.87 (2)	7(2.5)	7(2.5)	**3**(1)
Primary tumor	0.81 (3)	0.75 (2)	**0.71** (1)	0.15 (3)	0.34 (2)	**0.36** (1)	7(2)	11(3)	**4**(1)
Segment	**0.06** (1)	0.08 (2.5)	0.08 (2.5)	**0.94** (3)	0.92 (1.5)	0.92 (1.5)	33(3)	18(1)	**20(2)**
Sick	**0.06** (1.5)	0.07 (3)	**0.06** (1.5)	0.91 (3)	0.94 (2)	**0.97** (1)	7(1)	20(3)	**8**(2)
Sonar	0.29 (3)	0.24 (2)	**0.22** (1)	**0.70** (1)	0.58 (3)	0.62 (2)	9(3)	8(2)	**5**(1)
Vote	0.03 (3)	**0.02** (1.5)	**0.02** (1.5)	0.97 (3)	**0.98** (1.5)	**0.98** (1.5)	**3**(1)	**3**(1)	**3**(1)
Waveform 5000	**0.24** (1)	0.27 (2.5)	0.27 (2.5)	**0.76** (1)	0.73 (2.5)	0.73 (2.5)	143(3)	19(2)	**18**(1)
Wbdc-mod2	**0.06** (1)	0.11 (3)	0.09 (2)	**0.94** (1)	0.89 (3)	0.91 (2)	7(2.5)	7(2.5)	**3**(1)
Wine	0.15 (3)	0.05 (2)	**0.02** (1)	0.84 (3)	0.95 (2)	**0.98** (1)	**7**(1)	**7**(1)	**7**(1)
Wine-white	0.48 (3)	**0.47** (1.5)	**0.47** (1.5)	0.49 (3)	**0.56** (1.5)	**0.56** (1.5)	205(3)	11(1.5)	**11**(1.5)
Wine-red	0.44 (3)	**0.41** (1.5)	**0.41** (1.5)	0.55 (3)	**0.60** (1.5)	**0.60** (1.5)	53(3)	19(2)	**14**(1)
Average ranks	2.35	1.75	**1.5**	2.62	1.87	**1.42**	2.08	2.23	**1.35**

For further analysis, this study ranked the algorithms according to their performance. The lowest rank was assigned to the best performing algorithm. Wherever there was a tie between any two algorithms, an average of the two ranks was assigned to both. The *HEARpart* approach attained the lowest average rank for all three evaluation metrics. The average ranks are given in the last row of Table 5.

To exclude the possibility of *HEARpart* being better by sheer chance factors, the results were further analyzed with the help of the Friedman Rank Sum test, which is a nonparametric test for comparing multiple algorithms on multiple datasets (Demsar, 2006). The null hypothesis in the Friedman test is that no algorithm performs significantly differently. The results of the Friedman test for the three decision tree classifier algorithms are summarized in Table 6. The results from Table 6 rejected the null hypothesis for all three evaluation metrics. This indicated that the performance of at least one of the algorithms was significantly different from the others.

Table 6

Friedman Test Results

Accuracy	F-measure	Tree size (# nodes)
Friedman test	Friedman test	Friedman test
Ch-squared value=9.74	Ch-squared value=17.65	Ch-squared value=14.97
P-value=0.008	P-value=0.00016	P-value=0.0006
Outcome of the test: Null hypothesis is rejected	Outcome of the test: Null hypothesis is rejected	Outcome of the test: Null hypothesis is rejected

Next, for the pair-wise comparison of algorithms, Nemenyi post-hoc test by Demsar (2006) was applied and the results are summarized in Table 7. Based on the resulting p-values from the Nemenyi test, it can be said that *HEARpart* was a winner algorithm in several ways. It could construct simpler decision tree classifiers with less error rate and high F-measure as compared to other algorithms considered in the study.

Table 7

Pair-wise Comparison of the Performance of Decision Tree Classifiers

Evaluation metric	Nemenyi test p-values			Summary of the result
Accuracy		J48	*Rpart*	The *HEARpart* method is significantly better than the J48 classifier at a significance level of 0.05 (5%).
	Rpart	0.24	-	
	HEARpart	**0.013**	0.437	
F-measure		J48	*Rpart*	The *Rpart* classifier is significantly better than the J48 classifier at a significance level of 0.1 (10%). The *HEARpart* method is significantly better than the J48 classifier at a significance level of 0.05 (5%).
	Rpart	**0.084**	-	
	HEARpart	**0.00032**	0.189	
Tree size (Total nodes)		J48	*Rpart*	The *HEARpart* method is significantly better than the J48 classifier at a significance level of 0.05 (5%). The *HEARpart* method is significantly better than the *Rpart* classifier at a significance level of 0.05 (5%).
	Rpart	0.83		
	HEARpart	0.0184	0.003	

Table 8

Comparison of HEARpart and HEAD-DT

Dataset	Error		F-measure		Tree size	
	HEAD-DT	*HEARpart*	HEAD-DT	*HEARpart*	HEAD-DT	*HEARpart*
Abalone	0.80	**0.76**	0.20	**0.34**	4068	**8**
Audiology	**0.20**	0.48	**0.79**	0.68	119	**9**
Bridges version1	**0.40**	0.46	0.56	**0.64**	157	**5**
Car	**0.02**	0.07	**0.98**	0.92	172	**32**
Cylinder bands	**0.28**	0.33	**0.72**	0.70	211	**14**

(continued)

Dataset	Error		F-measure		Tree size	
	HEAD-DT	*HEARpart*	HEAD-DT	*HEARpart*	HEAD-DT	*HEARpart*
Glass	**0.27**	0.40	**0.72**	0.65	86	**13**
Segment	**0.03**	0.08	**0.97**	0.92	133	**18**
Sick	**0.01**	0.06	**0.99**	0.97	154	**20**
Wine-red	**0.36**	0.41	**0.63**	0.60	796	**19**
Wine-white	**0.37**	0.47	**0.63**	0.56	2526	**11**

Tables 9 and 10 report the performance comparison of *HEARpart* to the research works of Otero et al. (2012) and Karabadji et al. (2017). The results in Table 9 implied that the ant miner algorithm for decision tree construction had less error rates. *HEARpart* evolved comprehensible trees of smaller sizes across all datasets in comparison to the ant miner. According to Table 10, the ES achieved less error rate for most of the datasets. Going by F-measure, the performance of *HEARpart* and ES was comparable since each of the algorithms dominated in six datasets. *HEARpart* stood as a clear winner in terms of the size of decision trees across all datasets.

Table 9

Comparison of HEARpart and Ant Miner

Dataset	Classification error		Tree size	
	ATM	*HEARpart*	ATM	*HEARpart*
Breast cancer	**0.27**	**0.27**	10	**5**
Breast cancer Wisconsin	**0.06**	0.30	9	**6**
Credit data	**0.14**	0.18	30	**9**
Dermatology	**0.06**	**0.06**	21	**11**
E. coli	**0.16**	0.21	16	**10**
Glass	**0.29**	0.40	20	**11**
Hepatitis	**0.18**	0.19	8	**3**
Parkinson's	**0.08**	0.14	8	**3**
Soybean	0.13	**0.12**	50	**32**
Vote	0.05	**0.02**	6	**3**
Wine	0.04	**0.02**	**6**	7

Table 10

Comparison of HEARpart and ES for Decision Tree Construction

Dataset	Error		F-measure		Tree size	
	ES	*HEARpart*	ES	*HEARpart*	ES	*HEARpart*
Abalone	**0.76**	**0.76**	0.12	**0.34**	1245	**8**
Breast cancer Wisconsin	**0.01**	0.03	**0.98**	0.97	25	**8**
Breast cancer	**0.11**	0.27	0.56	**0.95**	21	**12**
Dermatology	**0.00**	0.06	**1.00**	0.92	15	**11**
Diabetes	**0.20**	0.24	**0.76**	0.72	83	**24**
E. coli	**0.12**	0.21	0.49	**0.81**	**11**	**11**
Hepatitis	0.07	**0.19**	0.82	**0.90**	11	**5**
Parkinson's	**0.00**	0.14	**1.00**	0.87	17	**7**
Primary tumor	**0.61**	0.71	0.18	**0.36**	83	**11**
Sonar	**0.15**	0.22	**0.84**	0.62	21	**8**
Soybean	0.07	**0.12**	0.85	**0.89**	89	**32**
Waveform 5000	**0.21**	0.27	**0.78**	0.73	429	**19**

CONCLUSION

Constructing optimal decision trees for datasets over different data domains is an arduous task. The performance of a decision tree model depends upon the settings of its hyper design parameters. A set of values for hyper parameter that is successful for one dataset may not give the expected performance on the other datasets. This paper proposed a hyper-heuristic evolutionary approach (*HEARpart*) for tuning hyper parameters of decision tree models to appropriate values during the construction phase. The suggested approach tunes the four important hyper parameters whose value may vary from one dataset to another. These include the minimum number of instances required at a node for inducing a split, depth of the tree, node splitting criterion, and complexity factor to control the amount of pruning. These four parameters control the structure of the decision tree models to a very large extent. The present approach is capable of inherently finding a decision tree classifier that is optimally tuned to the characteristics of the underlying training data.

Empirical evidence obtained in this paper showed that the *HEARpart* technique generated more accurate and comprehensible decision tree classifiers as compared to *Rpart* and WEKA's J48. When compared to other similar research works, *HEARpart* produced significantly comprehensible decision tree models, which resulted in a slight compromise on accuracy. It is understandable given that accuracy and comprehensibility are two conflicting criteria. The contribution of the proposed approach is that it liberates the research field from changing the important hyper parameters for decision tree construction of each dataset individually in obtaining optimal classifiers. The approach suggested in this study can also be tested for constructing regression trees in the future. Since accuracy and comprehensibility are the two competing performance metrics, a multi-objective hyper-heuristic approach for constructing decision trees needs to be investigated.

ACKNOWLEDGMENT

This research received no specific grant from any funding agency in the public, commercial, or non-profit sectors**.**

REFERENCES

Adibi, M. A. (2019). Single and multiple outputs decision tree classification using bi-level discrete-continues genetic algorithm. *Pattern Recognition Letters*, *128*, 190–196. https://doi.org/10.1016/j.patrec.2019.09.001

Barros, R. C., Basgalupp, M. P., de Carvalho, A. C. P. L. F., & Freitas, A. A. (2012). A hyper-heuristic evolutionary algorithm for automatically designing decision-tree algorithms. In *Proceedings of the Fourteenth International Conference on Genetic and Evolutionary Computation Conference - GECCO* (pp. 1237–1244). https://doi.org/10.1145/2330163.2330335

Barros, R. C., Basgalupp, M. P., Freitas, A. A., & de Carvalho, A. C. P. L. F. (2014). Evolutionary design of decision-tree algorithms tailored to microarray gene expression data sets. *IEEE Transactions on Evolutionary Computation*, *18*(6), 873–892. https://doi.org/10.1109/tevc.2013.2291813

Bharadwaj, K. K., & Saroj. (2009). Parallel genetic algorithm approach to automated discovery of hierarchical production rules. In J. Mehnen, M. Koppen, A. Saad, & A. Tiwari (Eds.), *Applications of Soft Computing* (pp. 327–336). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-540-89619-7_32

Bharadwaj, K. K., & Saroj. (2010). A parallel genetic programming based intelligent miner for discovery of censored production rules with fuzzy hierarchy. *Expert Systems with Applications*, *37*(6), 4601–4610. https://doi.org/10.1016/j.eswa.2009.12.048

Breiman, L., Friedman, J., Stone, C. J., & Olshen, R. A. (1984). *Classification and regression trees* (31). CRC press.

Cha, S.-H., & Tappert, C. (2009). A genetic algorithm for constructing compact binary decision trees. *Journal of Pattern Recognition Research*, *4*(1), 1–13. https://doi.org/10.13176/11.44

Cutler, A., Cutler, D. R., & Stevens, J. R. (2012). Random Forests. In C. Zhang & Y. Ma (Eds.), *Ensemble machine learning: Methods and applications* (pp. 157–175). Springer US. https://doi.org/10.1007/978-1-4419-9326-7_5

Demsar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine Learning Research*, *7*, 1–30. https://dl.acm.org/doi/10.5555/1248547.1248548

Drake, J. H., Kheiri, A., Özcan, E., & Burke, E. K. (2020). Recent advances in selection hyper-heuristics. *European Journal of Operational Research*, *285*(2), 405–428. https://doi.org/10.1016/j.ejor.2019.07.073

El Yafrani, M., Martins, M., Wagner, M., Ahiod, B., Delgado, M., & Luders, R. (2018). A hyperheuristic approach based on low-level heuristics for the travelling thief problem. *Genetic Programming and Evolvable Machines*, *19*(1–2), 121–150. https://doi.org/10.1007/s10710-017-9308-x

Esposito, F., Malerba, D., Semeraro, G., & Kay, J. (1997). A comparative analysis of methods for pruning decision trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *19*(5), 476–491. https://doi.org/10.1109/34.589207

Fu, Z., Golden, B. L., Lele, S., Raghavan, S., & Wasil, E. A. (2003). A genetic algorithm-based approach for building accurate decision trees. *INFORMS Journal on Computing*, *15*(1), 3–22. https://doi.org/10.1287/ijoc.15.1.3.15152

Han, J., Pei, J., & Kamber, M. (2011). *Data mining: Concepts and techniques*. Elsevier.

Hemmateenejad, B., Shamsipur, M., Zare-Shahabadi, V., & Akhond, M. (2011). Building optimal regression tree by ant colony system–genetic algorithm: Application to modeling of melting points. *Analytica Chimica Acta*, *704*(1–2), 57–62. https://doi.org/10.1016/j.aca.2011.08.010

Karabadji, N. E. I., Khelf, I., Seridi, H., & Laouar, L. (2012). Genetic optimization of decision tree choice for fault diagnosis in an industrial ventilator. In T. Fakhfakh, W. Bartelmus, F. Chaari, R. Zimroz, & M. Haddar (Eds.), *Condition monitoring of machinery in non-stationary operations* (pp. 277–283). Springer. https://doi.org/10.1007/978-3-642-28768-8_29

Karabadji, N. E. I., Seridi, H., Bousetouane, F., Dhifli, W., & Aridhi, S. (2017). An evolutionary scheme for decision tree construction. *Knowledge-Based Systems*, *119*, 166–177. https://doi.org/10.1016/j.knosys.2016.12.011

Karabadji, N. E. I., Seridi, H., Khelf, I., Azizi, N., & Boulkroune, R. (2014). Improved decision tree construction based on attribute selection and data sampling for fault diagnosis in rotating machines. *Engineering Applications of Artificial Intelligence*, *35*, 71–83. https://doi.org/10.1016/j.engappai.2014.06.010

Liu, D., & Fan, S. (2014). A modified decision tree algorithm based on genetic algorithm for mobile user classification problem. *The Scientific World Journal*, *2014*, 1–11. https://doi.org/10.1155/2014/468324

Mantovani, R. G., Horvath, T., Cerri, R., Vanschoren, J., & de Carvalho, A. C. P. L. F. (2016). Hyper-parameter tuning of a decision tree induction algorithm. In *5th Brazilian Conference on Intelligent Systems (BRACIS)* (pp. 37–42). https://doi.org/10.1109/bracis.2016.018

Michalewicz, Z. (1996). *Genetic algorithms + data structures = evolution programs* (3rd ed.). Springer-Verlag. https://doi.org/10.1007/978-3-662-03315-9

Otero, F. E. B., Freitas, A. A., & Johnson, C. G. (2012). Inducing decision trees with an ant colony optimization algorithm. *Applied Soft Computing*, *12*(11), 3615–3626. https://doi.org/10.1016/j.asoc.2012.05.028

Pacheco, J., Alfaro, E., Casado, S., Gamez, M., & García, N. (2012). A GRASP method for building classification trees. *Expert Systems with Applications*, *39*(3), 3241–3248. https://doi.org/10.1016/j.eswa.2011.09.011

Polikar, R. (2012). Ensemble Learning. In C. Zhang & Y. Ma (Eds.), *Ensemble machine learning: Methods and applications* (pp. 1–34). Springer US. https://doi.org/10.1007/978-1-4419-9326-7_1

Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning*, *1*(1), 81–106. https://doi.org/10.1007/BF00116251

Sabar, N. R., Turky, A., Song, A., & Sattar, A. (2017). Optimising deep belief networks by hyper-heuristic approach. In *IEEE Congress on Evolutionary Computation (CEC)* (pp. 2738–2745). https://doi.org/10.1109/cec.2017.7969640

Stein, G., Chen, B., Wu, A. S., & Hua, K. A. (2005). Decision tree classifier for network intrusion detection with GA-based feature selection. In *Proceedings of the 43rd Annual Southeast Regional Conference - Volume 2* (pp. 136–141). https://doi.org/10.1145/1167253.1167288

Therneau, T. M., Atkinson, E. J., & Foundation, M. (2019). *An introduction to recursive partitioning using the rpart routines*. 60.

Witten Ian H., & Frank, E. (2011). *Data mining: Practical machine learning tools and techniques*. Elsevier. https://doi.org/10.1016/C2009-0-19715-5

Yu, X., & Gen, M. (2010). *Introduction to evolutionary algorithms*. Springer-Verlag. https://doi.org/10.1007/978-1-84996-129-5