International Journal of Engineering and Advanced Technology (IJEAT)
ISSN: 2249-8958 (Online), Volume-8 Issue-4, April 2019

Forays into Decision Tree Learning Methods Review of Methods and Tools

Pravin S. Game, Vinod Vaze, Emmanuel M.

Abstract: ***One of the popular methods for supervised learning is decision trees. It has gained its popularity being a simple to use, easy to understand, and having no need to make any prior assumptions about the data. Decision trees have also achieved veracity of usage as it can be used to construct models for both numerical as well as categorical data. Numerous research studies have been done on the decision trees. This paper aims to study the methods used to construct decision trees, each one having its own significance. This work is focused on surveying these works. It not only covers the popular construction algorithms but also some advanced algorithms and the algorithms present in the commonly used academic research tool. Various induction mechanisms are studied based on the literature from standard publications, well known in the academic and research communities. The methods studied show that the respective algorithms have some pros as well as cons. The algorithms in the tools are tested on a standard dataset to predict the heart disease. The results show that though the same method is used as basis for construction of decision trees in two different tools, the results are quite different. This work, wherein more than 20 algorithms are touched upon and 5 tools are briefed, also helps to understand the overall evolution of the learning strategies for decision tree constructions.***

Index Terms**: Decision trees, learning, data classification, induction, optimization.**

I. INTRODUCTION

'Divide and conquer' methodology has been used to solve lot of problems in various fields of research. A decision tree algorithm also uses this strategy to create a hierarchical structure. Decision trees are considered to be one of simplest and easiest classifiers available. Given a dataset decision trees applies divide and conquer strategy to create a tree like structure with root at the top of hierarchy and leaves at the bottom. Applying the decision tree for classification of a dataset is analogous to following a flowchart. Each non-leaf node represents the divide strategy, meaning a question is asked and based on that the data is partitioned. At the next level, another question is asked based on the local dataset now available after the previous partition. It is continued till either no further division is possible or required set of records is represented by the leaf node or required number of divisions are performed [1-5]. This can be well illustrated using an example.

Consider a dataset of diseases with various attributes. The objective is to find whether a person is suffering from gaming addiction (digital or video games), in least number of steps. The gaming addiction is recently characterized and added as a disease by World Health Organization (WHO) in the 11th edition of International Classification of Diseases [6].

The root node partitions the data into two major categories based on whether the person plays digital games or not. So here decision parameter or division parameter is 'plays games'. Records where answer is 'Yes', i.e. all the people who play games is then further divided by checking whether the person has control over gaming or not. This will divide the records again into two categories. All the records saying the person has no control on gaming is then tested for priority i.e. whether person given priority to gaming or other activities. All the records where priority is given to gaming activity are then tested for whether the person continued or escalated playing games after going through negative consequences.

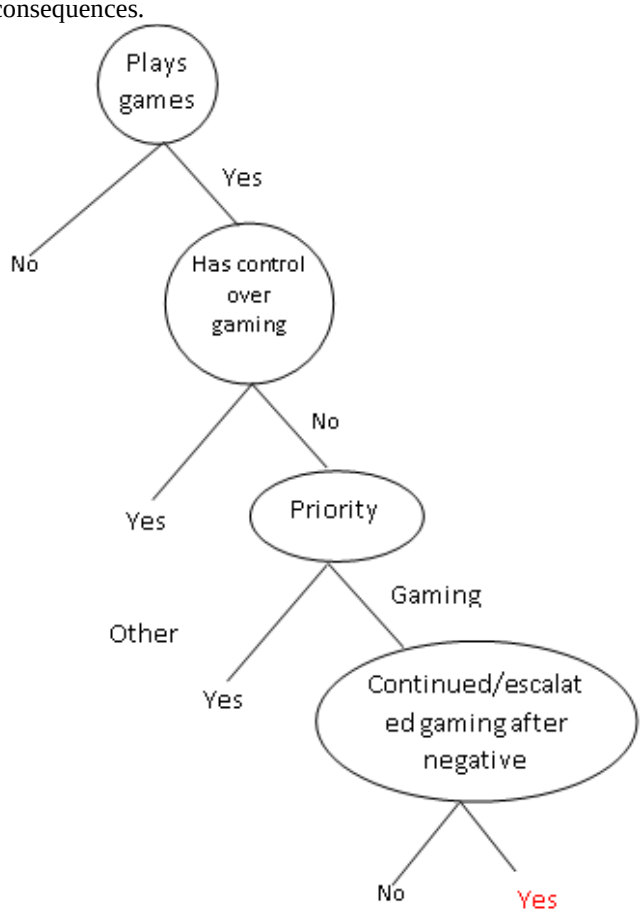

Fig. 1 Decision tree for classifying a gaming-addicted person

Manuscript published on 30 April 2019.
* Correspondence Author (s)
Pravin S. Game*, Research Scholar, JJT University, Jhunjhunu, Rajasthan, India.
Dr. Vinod Vaze, JJT University, Jhunjhunu, Rajasthan, India.
Dr. Emmanuel M., Pune Institute of Computer Technology, Pune, India.

If answer to this question is yes, the decision tree says that the person is suffering through gaming addiction. The tree growth is greedy in nature. The constructed decision tree is represented in Fig. 1.

It is easy to transform the tree into rules. This can be done even by using simple if-then statement [5]. For the example stated above, the final rule can be:

If (Person Plays_game= Yes && Has_control_on_gaming = No && Priority = gaming && Continued_gaming_after_negative_consequences = Yes) Then gaming_addiction = Yes

It can be observed that it is very easy and simple to build the decision tree as well as to understand. The internal nodes starting from root node provides a set of rules to reach a decision. What is important here is how to select these rules so that the tree height is least possible and this requires machine learning. Selection of the attribute will affect how many steps are required to reach the final decision, where no further classification is possible or required.

In machine learning, there are various types of leaning viz. supervised, unsupervised, reinforced. Decision trees fall in the category of supervised learning. Supervised learning is used to build the classification model to classify given data or dataset. In supervised learning the system is trained using historical datasets available and this dataset is a labeled dataset. This data is collection of individual instances also called as records. For the dataset, there are certain characteristics that are representative of the quality of each instance or record. These characteristics are generally called as attributes. Based on the attribute values each record is given a label or a class, hence, it is called as labeled dataset. A subset of this dataset is used as training data for the construction algorithm to construct the decision tree. The model is build using this training data. The model thus build is then validated using the remaining subset of the dataset called as testing data for generating the output labels for each record. The training as well as testing data contains the records which are considered to be correctly classified or labeled. Thus, when the model is applied on the testing dataset, the results of the classification can be compared with the correct classification label already applied to the records. This helps to understand and predict the performance of the constructed model on completely new and unseen records. [2-5]

In decision trees, for partitioning or splitting the training data at each node, attributes are used. The selection of the attributes will help to formulate the rules. There are measures to identify how good the split is. The goodness of the partition can be measured by using impurity index i.e. entropy, information gain or gini index. The purity refers to the classification of all the records, after applying the splitting criteria, branching out into a sub-subset belong to the same class [4-5]. Regardless of the impurity the tree construction can be stopped. Often to remove the redundancy some subtrees are removed or replaced. This is generally called as pruning. If pruning is done before the tree becomes full, it is called as prepruning. Whereas, if the pruning is applied after tree construction is complete, it is called as postpruning. Pruning improves the performance of the tree as redundant comparisons are removed [3]. Pruning is the process which is reverse of splitting.

Basically there are two types of decision trees:

- Classification trees : it is a decision tree whose output is a class or classes to which the given data belongs. So at the root node we have entire dataset, which gets divided and subdivided as decision rules are applied at each subsequent node of the tree. This data is generally discrete.
- Regression trees: it is a decision tree whose output is a real number. This is generally a continuous value.

The main interest of this paper is classification. In general, let is assume that a dataset to be classified consists of n instances and m attributes; and the instances are quite different from each other. In such case to build tree with n leaves and depth of O(log n) will have a computational cost as O(mn log n). In cases where some attributes are numerical sorting will be required which will cost, in worst case, O(n log n); however this does not change the cost of tree. In cases where pruning is used by subtree replacement, considering the cost of reclassification due to lifting of subtree between its root and leaf nodes will have a cost of $O(n (\log n)^2)$. This is additional cost and the total cost of building decision tree comes out to be

$$O(mn \log n) + O(n (\log n)^2).$$

As said earlier, selection of the attribute affects the performance of the decision tree. The optimality of decision trees generally refers to minimum number of levels or decision rules to represent the maximum classified data. To have such optimal decision trees there are various algorithms developed over the time, which are briefed in the next section.

The major objective of this work is to survey the foremost and classic methods available in the literature for classifying a dataset using decision trees. Another objective is to study various tools available which use many of these methods for classifying the datasets. The paper is organized as follows: the next section presents the survey of various decision tree learning methods; section III presents the tools available for such classification including the experimentation and the results are presented in section IV; section V is dedicated to discussions followed by conclusions in section VI.

II. DECISION TREE CONSTRUCTION METHODS

Abundant research has been done in the past since 1962 and is still being done to provide learning to decision trees. As decision tree is constructed in top-down fashion the algorithms are generally grouped in a family labeled as top-down induction of decision trees. The decision trees could be binary or multiway; binary infers that each node is further split into two subnodes, whereas, multiway means that the splitting results into more than two subnodes.

Following are the algorithms used to build and/or optimize decision trees found in the literature. There are quite a few which are popular with the researchers while others are not so popular but advanced decision trees algorithms.

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication (BEIESP)

A. CLS

Concept Learning Systems (CLS) [7] can be called as the first attempt in learning for classification. Concept can be a thought or opinion. Concept learning is defined as learning about certain entity to understand what the entity is in general. Classification depends on the degree to which the concept is learned. Concept learning is defined as a term which applies to any situation in which a subject learns to make an identifying response to members of a set of not completely identical stimuli, subject to the following restrictions:

1. The subject must, conceivably, be able to instruct a human to apply the classification rule. The subject is not allowed to use examples during the course of this instruction.
2. The rule to be learned must be one that can be applied to any appropriate stimulus regardless of the context in which the stimulus appears.
3. The rule must be deterministic; once a given stimulus is completely described it must be uniquely classifiable.

This work has been accepted as opening learning system for subsequent decision tree learning system.

B. THAID

This algorithm has been proposed as a successor program to Automatic Interaction Detector (AID) computer program [8] for classification of data. Actually the decision trees for statistical analysis were developed for the analysis of survey data. AID formed the foundation for analyzing the large survey data with multiple variables. In this approach Theta criteria is added to classify data and provide categorical output, hence the name THAID (THeta AID). Earlier it was developed for the independent variables. Subsequently the usability of THAID is also proved for the analysis of nominal scale dependent variables by having theta or delta criteria [9]. Here minimum theta or minimum delta criteria is taken from the user as input for splitting the data. The split is performed only if the values are more than this minimum. The algorithm terminates when this minimum criteria is no longer met indicating that no further splitting is permissible. The resultant groups at each level, when seen sequentially, represent a tree called as THAID tree. The implementation is found to give better results than the predecessor AID program. Around the same time when AID was being improved, Honeywell developed another program for survey analysis called MAID [10], which improved the AID for multivariate analysis.

C. CHAID

As an improvement or offshoot to AID and THAID, [11] developed a classification method based on Chi-square. This method is one of the well-known and most used methods for classification of data. Along with the monotonic predictors and free predictors, which are used in AID, this method introduces floating predictors. This floating predictor is used to deal with the missing information for some attributes in the data. The method uses p-value of Chi-square method. For each of the predictors Bonferroni multiplier is derived to identify the bounds for the significance level. It also introduced continual significance testing. Such testing at each stage of classification provided the spitting criteria for multiway subsequent classification of data. Such multiway subdivisions are sometimes more useful than just the binary splitting of the data at each stage. There are various other algorithms found in the literature with focus on building a multiway split decision trees viz. [12-18].

D. ID3

Iterative Dichotomiser 3(ID3) introduced in [19] [20] is successor of CLS system. It uses divide-and-conquer technique for building decision trees resulting into top down construction of the trees getting the name as top-down induction of decision trees and have a palindromic abbreviation TDIDT. This algorithm is third in a series of programs developed as a response to the challenge in a chess game. It is based on the CLS framework and uses information gain as a measure of impurity to decide the best splitting attribute. It selects the attributes having highest information gain for splitting. It is based on the Occam's principle which states that the simplest solution tends to be the right one. In the context of decision trees the ID3 algorithm tries to generate the tree with smallest possible height, which also emphases the choice of the test or attribute for splitting. This method results into binary splitting. This research acted as the foundational work for subsequent learning algorithms viz. ACLS[21], ASSISTANT[22], EX-TRAN[23], Rule Master[24]. However, this technique suffers when implemented on dataset having missing values for some records and information gain approach may lead to the problem of overfitting.

E. C4.5

To overcome the drawbacks of ID3, it was improved in a series of programs and the author presented this version called as C4.5 - a more robust, practical, refined and influential system [25]; which is available in the public domain for building the decision trees. Instead of using information gain for splitting attributes it uses gainratio which deals with the problem of overfitting. It has methods to replace the missing values in the records by predicting that value based on the value for that attribute in other records. C4.5 has proposed and implemented two pruning strategies viz. subtree replacement and subtree raising. This algorithm is one of the most used techniques for decision tree induction.

F. CART

Classification and Regression Trees (CART) developed by [26] is one of the most sought after method for classification. We have already defined the classification trees and regression trees in earlier section. It differs from ID3 and C4.5 in use of splitting criteria. Instead of information gain as a criterion for splitting a node, CART uses Gini index and for pruning of the tree it uses cross-validation [27]. This method learns more itself and turned out to be machine learning needing less input from the data analyst. However, it is slower sometimes due to complex calculation of Gini index used for splitting and finding the best split. CART generates a binary decision tree. The entire code of CART is proprietary [28]. Another approaches using minimum Gini index are SPRINT [29] and SLIQ [30] developed at IBM for classification of large datasets.

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication (BEIESP)

G. FACT

Fast Algorithm for Classification Trees, abbreviated as FACT, was developed by [31] and compared to CART. Though CART was very good at classification and addressed the drawbacks of its earlier methods, it still had scope for improvement. FAST combined the speed of Linear Discriminant Analysis (LDA) and visualization of decision trees of CART, known to be the best features of LDA and CART. The tree constructed by FACT can have multiple splits at the nodes rather than binary as in the case of CART. It uses a modified version of LDA as a splitting criteria, can handle missing information in the records by estimation. Whereas CART uses cross-validation as a stopping criteria, FAST stops when the there is a single class in the node with the user defined minimum sample size or the when node error rate is not decreasing with further splitting. Though CART and FACT have comparable accuracy, FACT is little faster than CART. However, the effectiveness of the FACT is questionable in some cases due to its direct stopping, as the bottom-up pruning is not feasible like CART and it has been observed that FACT is biased for categorical variables.

H. FIRM

Formal Inference-based Recursive Modeling (FIRM) [32] tried to address the bias problem by using Bonferroni estimate for selecting predictors to split a node alongwith piecewise constant regression. Whereas, as it had been the case that AID and CART produced binary splits FIRM splits the ordered data into ten subnodes initially and divides the categorical data into all the available categories. To terminate splitting of a node it uses Neyman-Pearson statistical inference, resulting into creation of trees by forward selection. Hence it does not need to prune the tree as compared to CART. To confirm that the splitting of the node is real FIRM uses formal statistical methods, i.e. for every node it applies a null hypothesis- the node is not divisible, and tests it using conservative significance levels. This leads to creation of smaller trees, helping in easy interpretation. Later on it was observed that the adjustments using Bonferroni estimate results into bias as the estimate may over-correct. Hence the FIRM tends to be biased while splitting the numerical data. Later on it was used on large datasets to analyze structure-activity dataset [33].

I. QUEST

Understanding the biased nature and linear computational complexity of the exhaustive search based classification trees in selecting the attribute for division, [34] proposed QUEST having negligible bias. The same authors had proposed FACT with less computational complexity; however it still suffers from variable selection bias for some categorical attributes. Hence QUEST – Quick, Unbiased, Efficient, Statistical Tree: retains the speed of FAST (Quick), pruning is possible (Efficient), negligible bias hence Unbiased, gives a binary tree. P-value is used for variable selection. For spitting it first divides the data into two superclasses by using two-means clustering algorithm and then applies a modified quadratic discriminant analysis to identify the split point for an ordered variable. For finding the split in case of the categorical variable, largest discriminant coordinate is used to find the discriminatory information useful for identifying the spacing and ordering of the values, which will a node into two (binary). This also deals with the missing information. For the example datasets used for verifying the algorithms, QUEST is found to be way better in speed than FACT and sometimes than exhaustive search methods; also it is has negligible bias compared to these methods.

J. CRUISE

After analyzing the biased behavior of the then existing methodologies towards numerical and/or categorical variables as well as the pros and cons of binary and multi-way partitioning, [35] proposed Classification Rule with Unbiased Interaction Selection and Estimation, abbreviated as CRUISE. For split selection it has used and improved the concepts borrowed from their earlier work viz. FACT, QUEST and GUIDE. It splits the nodes into multiple subnodes, reducing the effective height of the tree. . Using local interactions between the variables, it generates shorter trees. For pruning, it has used and improved the techniques of CART. Whereas, QUEST has negligible bias, practically CRUISE is free of any bias. It is also found to be as fast as CART and works better for missing values.

K. GUIDE

Generalized, Unbiased Interaction Detection and Estimation-abbreviated as GUIDE- was developed by [36] with a major objective to remove bias in the selection of variable for splitting and as an improvement to SUPPORT [37]. It employs piecewise constant and piecewise linear model to construct trees, leading to three variants of GUIDE, namely piecewise constant GUIDE, piecewise linear GUIDE and a model which uses both-piecewise constant with piecewise linear GUIDE. As the name suggests, it has negligible bias, identifies the interaction between local variables by using chi-square analysis. This interaction detection is done for pairs i.e. pairwise interaction detection and this method also includes categorical predictors which are absent in SUPPORT. In GUIDE, nine forms of chi-square are calculated- three to detect curvatures in variables, three to detect pairwise interaction of same type of variables and three to detect pairwise interaction between different types of variables. To eliminate the bias in variable selection GUIDE included bootstrapping. Bootstrap estimation is used only at root node and not at other subnodes to reduce the computational cost. The results of the implementation of GUIDE are found to be quite encouraging. The concepts from this method are further extended to build quantile regression trees [38] and classification trees without bias and multiway splitting-CRUISE which is explained above.

L. SUPPORT

Smoothed and Unsmoothed Piece-wise Polynomial Regression Trees was developed by [37] and compared its performance mainly with CART. SUPPORT is a nonparametric function estimation method wherein estimate consists of several pieces. Each piece is obtained by fitting a polynomial regression rather than a constant as used in CART.

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication (BEIESP)

This estimation can be smoothed or let unsmoothed depending on the requirement. In case smoothing is required, it is achieved by using weighted averaging method to connect various pieces. To select a variable for spitting the node positive and negative residuals are used. For partitioning SUPPORT uses cross-validation estimate. When compared to CART, this method generated shorter trees, i.e. trees with lower levels, which is one of the requirements for improving the interpretation of the trees. SUPPORT differs from CART in generating the final tree; in the sense, it does not use backward pruning to shorten the trees; instead it determines the tree size by using multi-step look-ahead stopping rule along with the cross-validation stopping rule of CART.
Following discussion introduces the classes of decision trees and various frameworks developed in each category:

M. Oblivious trees

Major decision tree induction methods follow top-down approach for the construction. An alternative, bottom-up approach based on a recursive and nondeterministic algorithm for supervised learning was proposed by [39] by using Oblivious Read-once Decision Graphs (OODG). For a n-categorization function, a decision graph is defined as a directed acyclic graph with a root (having indegree zero), exactly n-category nodes (with zero outdegree) and branching nodes (labeled with a variable and outgoing edges labeled by the domains of variables). The term read-once depicts that the variable (node) occurs at-most once in the travelled path. Oblivious decision graphs are leveled i.e. outgoing edges from a node at a level, when node is split, terminate at the next level. Same variable is used to label all the nodes at same level, defining total ordering on the variable. The number of nodes in graph represents the size of the OODG and the number of nodes at a level represents the width of that level. A variable instance is categorized by traversing the path from the root node to category nodes. Two algorithms are proposed and tested are bottom-up approach and the hill-climbing approach (resulting into HOODG) is used for construction of tree. The HOODG gave 100% accuracy on the test data. Compared to C4.5, OODGs gave better performance for certain problems like Monk's problem and parity. However, as hill-climbing is used, the approached suffered from the problem of global minimum. The HOODG was improved by adding Tie-breaking heuristics [40] resulting in better accuracy and learning rate. The authors subsequently also proposed a top-down approach for construction of OODG using entropy similar to C4.5 and it is called as Entropy-based Oblivious Decision Graphs (EODG) [41]. In EODG, after construction of oblivious decision tree, pruning is done bottom-up and merging of nodes in top-down. Around the same time, [42] also proposed an oblivious decision tree construction algorithm called OBLIVION. It is specifically proposed to address the issue of learning in the presence of irrelevant features. The algorithm is developed to improve upon the earlier oblivious decision tree algorithm presented is [43] called FOCUS and [44] and experimentally found to be better than its precursors.

N. Online decision trees

The methods for construction of decision trees discussed above use the data which is stored in memory, which is offline data collected over a period of time. With increase in use of social media as well as online commerce it has become necessary to take decisions based on the current or live posts or transaction. This has led to a completely new family of decision trees- called online decision trees (ODT) - where the construction of decision trees takes place based on the current record. ODT learns based on the real-time online inputs without regards to other conditions like underlying distribution. This kind of algorithms is useful when the training data keeps on changing as a result of addition, deletion or modification of data due to transactions. This category of algorithms is focused on classifying large datasets. Major methods in this category are SPRINT, SLIQ, ID4[45], ID5R[46,47], VFDT[48], CVFDT[49], UCVFDT[50], OADT[51], PLANET[52], SPIES[53], CLOUDS[54], pCLOUDS[55], BOAT[56], SPDT[57]. A comprehensive comparison of most of the methods is given in [58], hence not discussed here.

O. Oblique trees

The standard decision tree algorithms, like ID3, C4.5, use axis-parallel splitting approach. First instance of oblique decision tree construction can be found in CART, where linear combination is used to split the data. Another top-down approach to build oblique decision is tree is proposed by [59] called Linear Machine Decision Trees (LMDT), in which a node is split by training a linear machine. Then linear machine classifies an instance. A linear machine is a multiclass classifier as defined in [60]. Linear machine is trained at every node using the training data repeatedly until an optimal split is found. A simulated annealing based approach, in which an oblique hyperplane is randomly selected to split the data was introduced in [Heath] and called Simulated Annealing of Decision Trees (SADT). Due to random selection of hyperplane, SADT resulted into different decision tree each time. The algorithm is run multiple times on the same input dataset and smallest trees are retained. It was observed that the algorithm generated small and quite accurate decision trees but the efficiency is an issue. Based on the strengths of CART (Linear Machine), LMDT and SADT, research by [62] proposed Oblique Classifier 1 (OC1) which used deterministic hill-climbing approach and randomization. This approach is computationally efficient. Though it has six impurity measures (information gain, Gini index, the Towing rule, Max Minority, Sum minority, and Sum of variances), The Towing Rule was used as default measure for the experimentation. Experimental results observed that with increased randomization, tree size decreased and the accuracy increased. The OC1 variations found to be better as compared to other systems. This OC1 was extended later on by [63] with extensions using evolutionary strategies (OC1-ES), genetic programming (OC1-GP) and simulated annealing (OC1-SA). Recent research in the development of oblique decision trees is Fisher's tree [64], Geometric Decision Tree [65], HHCART [66].

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication (BEIESP)

III. DECISION TREE CONSTRUCTION TOOLS

Apart from the facility provided by programming languages like Java, Python, R to implement the decision trees, various tools like WEKA, KNIME, Salford Predictive Modeler, SAS University Edition, Qlik Sense are also available.

This section some decision tree algorithms available in frequently quoted and used data analysis tools.

A. Weka

The latest stable version is 3.8 [67]. Weka stands for Waikato Environment for Knowledge Analysis. It is an open source tool useful for data analysis from preparation to visualization. For constructing decision trees, it has

- Hoeffding tree (based on VFDT)
- J48 (based on C4.5)
- LMT (Logistic Model Tree)
- RandomForest
- Random Tree
- REPTree

B. KNIME Analytics platform

The Konstanz Information Miner is another open source data analytics platform available freely (http://www.knime.com). It has variants as per the requirements of the research community. For decision tree construction it has

- Decision tree Learner (based on C4.5 and SPRINT)
- Simple Regression Tree Learner (based on CART)

C. Salford Predictive Modeler®

This is proprietary tool available for data analysis by Salford Systems (now acquired by Minitab, Inc.). We have downloaded the trial version of the tool for use. It consists of the CART mentioned earlier in this paper with the proprietary code from the authors and is being enhanced (https://www.salford-systems.com/products/cart).

D. SAS University edition®

Data mining and analysis tools from SAS Research & Development are most sought after tools in various industries today. This free access edition of the proprietary tool has various procedures for creation of decision trees. Mainly we find following procs used in various SAS tools.

- Proc DTREE [68]
- Proc HPSPLIT [69]
- Proc ARBORETUM (mixes CHAID and CART) [70]

E. IBM® SPSS® Modeler

This is another proprietary tool available in the market for construction of decision trees is from IBM, and used by many companies. This tool allows using following algorithms [71] for tree construction

- C&R Tree
- CHAID
- QUEST
- C5.0

Interactive tree building allows creating trees for first three algorithms. Support for C5.0 is not provided.

F. GNU PSPP

This tool is open source counterpart of IBM's SPSS tool published by Free Software Foundation. The tool has no expiry and no bound on use cases. It can be easily used in graphical mode as well as command line mode.

IV. RESULTS

For experimentation on the decision tree induction, Cleveland [72] dataset from UCI is used. It is a multivariate dataset with 303 instances and 14 attributes, where 14th attribute is the predicted class for heart disease. The 14 attributes are age, sex, type of chest pain, resting BP, cholesterol, fasting blood sugar, resting ECG results, maximum heart rate, angina, depression, slope, colored vessels, thal, and diagnosis (prediction). There are five classes in which heart disease is classified. Processed dataset is used as input. A standalone desktop computer system having Intel i3 core, 4 GB RAM is used to for executing the experiments. The dataset is classified using various algorithms in the tools as well as implementation in python.

The results for various decision tree algorithms from Weka and KNIME tool as well as python implementation are shown in Table I, Table II, Table III and Table IV.

The results in Table I show that the time taken to build the model, irrespective of splitting criteria, is minimum for Random Tree (0.001 second) followed by J48 (0.01seconds), Hoeffding tree (0.03 seconds), and Random Forest (0.2 seconds). LMT is found to take maximum time of 0.7 seconds. With 10-fold cross-validation, accuracy for correctly classifying the given instances of Random forest is found to the best i.e. 57.09%; followed by LMT - 55.77%, Hoeffding tree - 54.45%, and for J48 – 52.47%. Accuracy using Random trees algorithm is lowest – 48.51%. When same data is used for training and testing, the python implementation (Table 3) gave 98% accuracy, irrespective of gini index or entropy measure used to split the node. Table IV shows the results of the decision tree classifier available in the KNIME tool. Results show that the tool performs better when the same dataset is used for training and testing as depicted in Fig. 2 than with the splitting criteria. Even in that case, it gave almost 9% better results when gini index is used as compared with the gain ratio criteria.

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication (BEIESP)

Table I. Classification using Decision tree algorithms in Weka with 10-fold cross-validation

Parameters used	Algorithms				
	Hoeffding tree	**J48**	**LMT**	**RandomForest**	**Random Tree**
Correctly Classified Instances	165	159	169	173	147
Incorrectly Classified Instances	138	144	134	130	157
Kappa statistic	0.1536	0.223	0.2704	0.2663	0.2123
Mean absolute error	0.2333	0.2105	0.1975	0.2021	0.2059
Root mean squared error	0.3611	0.4011	0.3278	0.321	0.452
Relative absolute error (%)	90.0341	81.2618	76.2092	78.0175	79.4706
Root relative squared error (%)	100.5116	111.6521	91.2438	89.3655	125.8236
Time taken to build model (in seconds)	0.03	0.01	0.73	0.23	0.001

Table II. Classification using Decision tree algorithms in Weka with 70-30 split criteria (70% training data, 30% test data i.e. 212 instances for training the model and 91 instances for testing the model)

Parameters used	Algorithms				
	Hoeffding tree	**J48**	**LMT**	**RandomForest**	**Random Tree**
Correctly Classified Instances	46	51	50	52	47
Incorrectly Classified Instances	45	40	41	39	44
Kappa statistic	0	0.3356	0.2999	0.3167	0.2762
Mean absolute error	0.2641	0.1883	0.2039	0.209	0.1934
Root mean squared error	0.3695	0.3819	0.3365	0.3279	0.4398
Relative absolute error (%)	100	71.2912	77.223	79.1464	73.2321
Root relative squared error (%)	100	103.3489	91.0536	88.745	119.0169
Time taken to build model (in seconds)	0.03	0.01	0.75	0.2	0.001

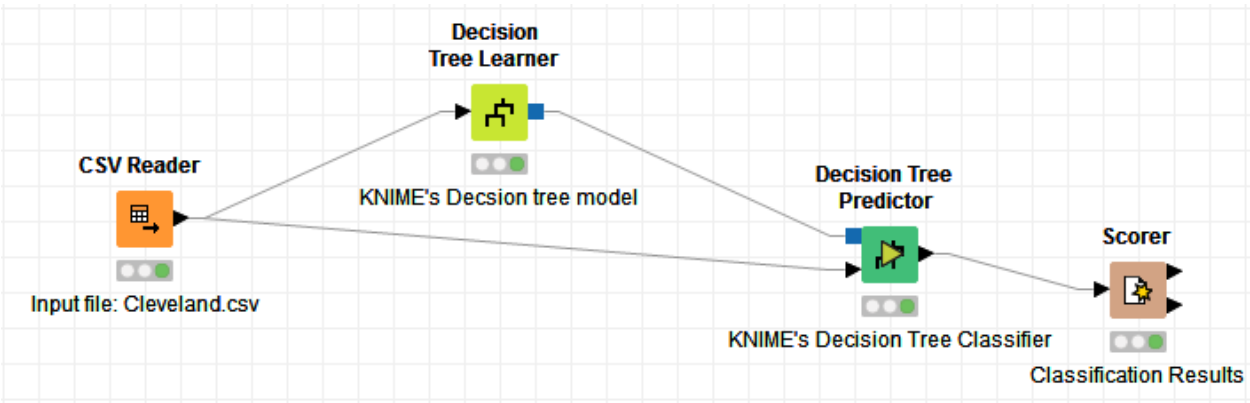

Fig. 2 KNIME Workflow for decision tree

Table III. Classification using Decision tree algorithms in Python

Parameters used	Function DecisionTreeClassifier available in integrated library			
	criterion = 'gini'		**criterion = 'entropy'**	
	Same Training and test data	70% training and 30 % testing data	Same Training and test data	70% training and 30 % testing data
Accuracy Score	98%	45%	98%	48%

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication (BEIESP)

Table IV. Classification using Decision tree algorithm in KNIME

Parameters used	KNIME Decision Tree			
	criterion = 'Gain Ratio'		criterion = 'Gini Index'	
	Same Training and test data	70% training and 30 % testing data	Same Training and test data	70% training and 30 % testing data
Overall Accuracy	58.1%	53.8%	67%	57.1%
Cohen's Kappa	0.188	0.21	0.443	0.262

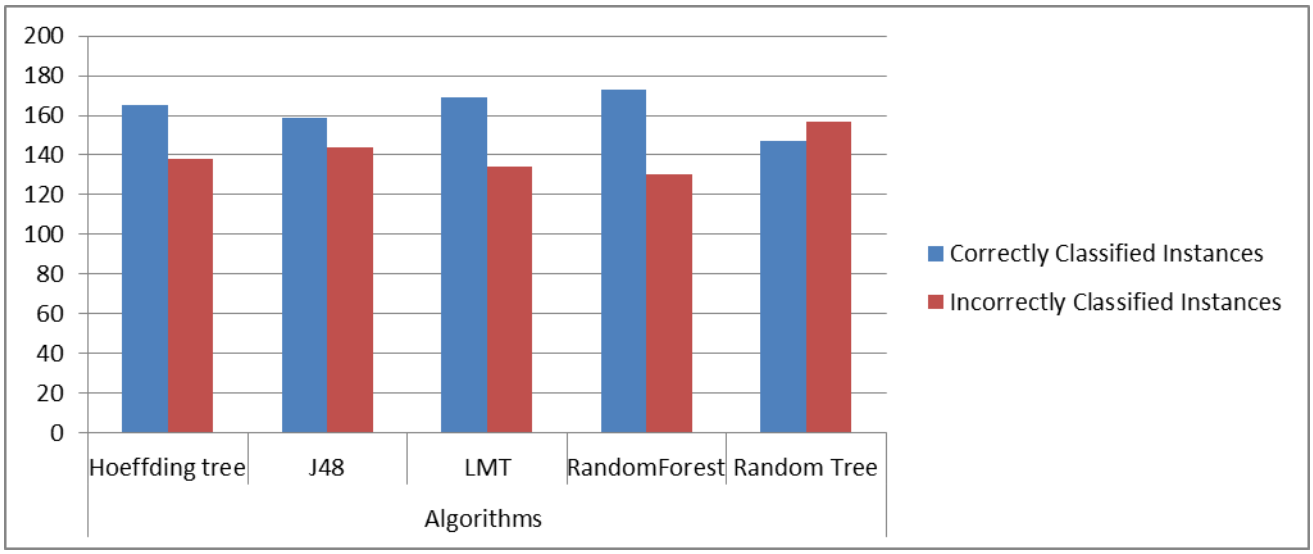

Fig. 3 Classification performances of respective algorithms from Table I with 10 fold cross-validation

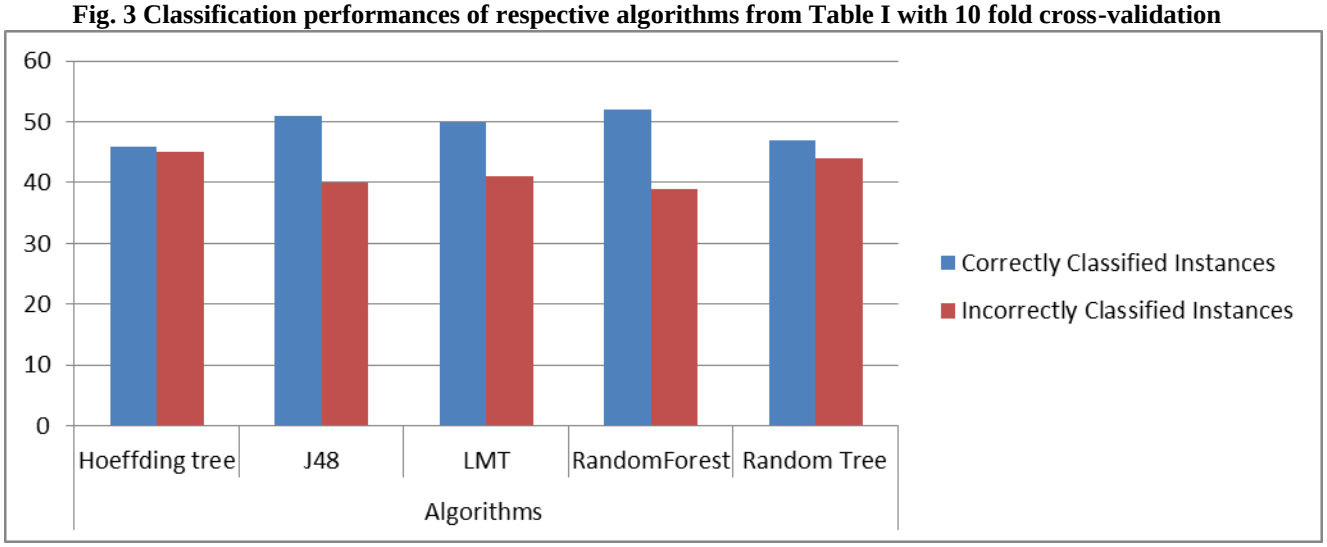

Fig. 4 Classification performances of respective algorithms from Table II with 70-30 splitting criteria

When the given dataset with 303 records is split using 70-30 criteria; 202 instances were used for training the model and 91 instances for testing the model. Time taken to build the model is same as already elaborated. It is observed that the accuracy of Random Forest is best -57.14% followed by J48 – 56.04%, LMT – 54.94%, Random trees – 51.64% and Hoeffding tree – 47.42 %. The python program classified the dataset with 45% accuracy with gini measure and with 48% accuracy with entropy measure.

V. DISCUSSION

The objective of the work was to understand various decision tree construction methods, their evolution over the time and test the functioning of algorithms on standard heart disease dataset to predict the disease. There is variety in induction methods based on the splitting criteria and some methods provide the facility of choosing the splitting criteria. The SPM tool provides the facility to choose one from seven different splitting criteria available. Whereas we also found that there are some distinct families of decision trees induction methods based on a concept like, oblivious trees or based on usage like, a separate set of research literature focused on scalable, large scale databases.Though two tools and a result from Python program are presented, the experiment was also done on few other tools.

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication (BEIESP)

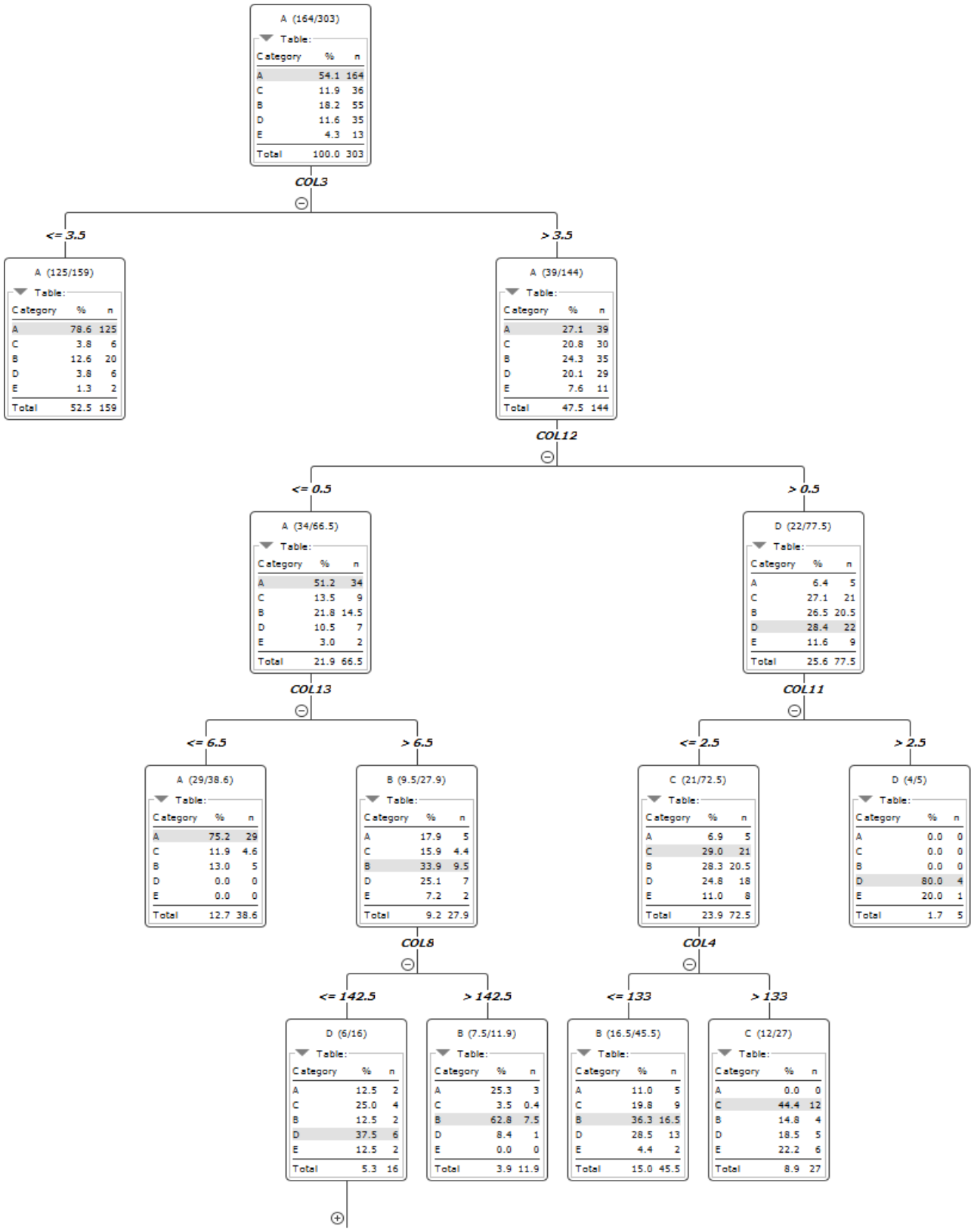

Fig. 5 Decision tree view from the KNIME tool

There are lots of facilities provided by not only the trial versions of the proprietary tools but also by the open source tool. However the selection of tool is the choice of the experimenter based on the requirements. Weka is found to be very good in providing statistical results whereas KNIME (Fig. 5) and SPM (evaluation version) give good graphical representations along with statistics.

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication (BEIESP)

Python program though gave output in console; using online tools the tree can be visualized. We used Graphviz Online for viewing the tree.

VI. CONCLUSION

Various types of decision trees based on certain criteria can be seen. Here, a comprehensive study of various foundational and recent decision tree construction algorithms is done. We have also identified the commonly used algorithms in the tools and performance of some algorithms using a dataset is presented. Heart disease prediction is quite useful for the early prediction of the disease which will assist in proactive measures to prevent the heart attack. Decision tree is a very good tool in assisting such prediction. Research in decision tree induction is an ongoing phenomenon. Various algorithms available in the tool have given considerable results. It can be observed that the implementation in python gave 98% accuracy when the system is trained and tested on the same data. Though this sounds exciting, this cannot be the true case always; as the train and test data is generally different. But it served a purpose to understand that the implementation is correct as the validation results are commanding with same input. However, the python implantation gave poor results when faced with standard split data. This shows that there is still scope for improvement. It is also observed that though algorithms are developed for various splitting criteria and some hybrid approaches are available, very little research is done using bio-inspired algorithms. This little use of bio-inspired algorithms is also true when considered for the large scale databases. In future, we plan to investigate use of a bio-inspired algorithm for construction of decision trees.

REFERENCES

1. I. H. Witten, E. Frank, M. A. Hall, *Data mining: practical machine learning tools and techniques*, Morgan Kaufmann, 2011, pp. 99-105.
2. F. Candy, *The data science handbook*, Wiley India, 2018, pp. 101-103.
3. E. Alpaydin, *Introduction to machine learning*, PHI Learning, 2010, pp. 185-202.
4. J. Bell, *Machine learning for big data: hands-on for developers and technical professionals*, Wiley India, 2015, pp. 45-65.
5. M. Geatz, R. Roiger, *Data mining: a tutorial based primer*, Pearson, 2003, pp. 31-38, 92-100.
6. WHO, *International Classification of diseases- 11th revision*, Released on 18th June 2018, www.who.int/classification/icd/en. Accessed on 20 September 2018.
7. E. B. Hunt, *Concept learning: an information processing problem*, (abstracts), John Wiley & Sons, 1962.
8. Sonquist JA, Baker EL, Morgan JN. Searching for structure (Alias-AID-III).Survey Research Center. Institute of Social Research University of Michigan, Ann Arbor 1971.
9. J. N. Morgan, R. C. Messenger, "THAID: a sequential analysis program for the analysis of nominal scale dependent variables", Survey Research Center, Institute of Social Research University of Michigan, Ann Arbor 1973.
10. M. W. Gillo, "MAID: a Honeywell 600 program for an automatised survey analysis", *Behavioral Science*, vol. 17, pp. 251-252, 1972
11. G. V. Kass, "An exploratory technique for investigating large quantities of categorical data", *Applied Statistics*, 1980; vol. 29, no. 2, pp. 119-127, 1980.
12. C. E. Brodley, "Automatic selection of split criterion during tree growing based on node location", *Proceedings of the Twelfth International Conference on Machine Learning, Tahoe City, California*, pp. 73-80, 1995.
13. U. M. Fayyad, K. B. Irani, "Multi-interval discretization of continuous-valued attributes for classification learning*", in Proceedings of Thirteenth International Joint Conference on Artificial Intelligence*, 1993, pp. 1022-1027.
14. T. Fulton, S. Kasif, S. Salzberg, "Efficient algorithms for finding multi-way splits for decision trees", *in Proceedings of the Twelfth International Conference on International Conference on Machine Learning, Tahoe City, California*,1995, pp. 244-251.
15. E. Frank, I. H. Witten, "Selecting multiway splits in decision trees", Department of Computer Science University of Waikato. 1996.
16. T. Elomaa, J. Rousu, "Finding Optimal Multi-Splits for Numerical Attributes in Decision Tree Learning", *Produced as part of the ESPRIT Working Group in Neural and Computational Learning*, NeuroCOLT Technical report, NC-TR-96-041, 1996.
17. H. Kim, W. Y. Loh, "Classification trees with unbiased multiway splits", *Journal of the American Statistical Association*, vol. 96, no. 454, pp. 589-604, 2001.
18. F. Berzal, J. C. Cubero, N. Marin, D. Sanchez, "Building multi-way decision trees with numerical attributes", *Information Sciences*, vol. 165, no. 1-2, pp. 73-90, 2004.
19. J. R. Quinlan, "Learning efficient classification procedures and their application chess end games", *In Machine Learning- An Artificial Intelligence approach*, ed. R. S. Michalski, J. G. Carbonell, T. M. Mitchell, pp. 463-482, vol. 1, 1983.
20. J. R. Quinlan, "Induction of decision trees", *Machine Learning*, vol. 1, pp. 81-106, 1986.
21. A. Patterson, T. Niblett, "ACLS user manual", Glasgow: Intelligent Terminals Ltd. 1983.
22. I. Kononenko, I. Bratko, E. Roskar, "Inductive learning system ASSISTANT", *Informatica*, vol . 9, no. 3, pp. 44-55, 1985.
23. T. Hassan, M. A. Razzak, D. Michie, "EX-TRAN 7—a different approach for an expert system generator", *in Proceedings of Fifth international workshop on Expert systems & their applications*, pp. 153-170, volume 1. 1985, ACM.
24. C. E. Riese, J. D. Stuart, "A Knowledge-Engineering Facility for Building Scientific Expert Systems", *in Artificial Intelligence Applications in Chemistry, ACS Symposium Series*, pp. 18-30, vol.306, 1986.
25. J. R. Quinlan, "C4.5: Programs for machine learning", *Morgan Kaufmann*, 1993.
26. L. Breiman, J. Friedman, J. C. Stone, R. A. Olshen, "Classification and Regression Trees", *Taylor & Francis*, 1984.
27. Popular Decision Tree: Classification and Regression Trees (C&RT) http://www.statsoft.com/Textbook/Classification-and-Regression-Trees . Accessed 25 September 2018.
28. CART®-Classification and Regression Trees. https://www.salford-systems.com/products/cart Accessed 25 September 2018.
29. J. Shafer, R. Agrawal, M. Mehta, "SPRINT: A scalable parallel classifier for data mining*", in proceedings of the 22nd VLDB conference, Mumbai, India*, pp. 544-555, 1996.
30. M. Mehta, R. Agrawal, J. Rissanen, "SLIQ: A fast scalable classifier for data mining", *in Fifth International conference on extending database technology. In Apers P., Bouzeghoub M., Gardarin G. (eds) Advances in Database Technology — EDBT '96. Lecture Notes in Computer Science*, vol. 1057, pp. 18-32, 1996.
31. W. Y. Loh, N. Vanichsetkul, "Tree-structured classification via generalized discriminant analysis", *Journal of the American Statistical Association*, vol. 83, no. 403, pp. 715-725, 1988.
32. D. M. Hawkins, "FIRM-Formal inference-based recursive modelling", *Technical report number 546*, University of Minnesota, 1990.
33. D. M. Hawkins, S. S. Young, A. Rusinko III, "Analysis of a large structure-activity data set using recursive partitioning", *Molecular Informatics*, vol. 16, no. 4, pp. 296-302, 1997.
34. W. Y. Loh, Y. S. Shih, "Split selection methods for classification trees", *Statistica Sinica*, vol. 7, pp. 815-840, 1997.
35. H. Kim, W. Y. Loh, "Classification trees with unbiased multiway splits", *Journal of the American Statistical Association*, vol. 96, no. 454, pp. 589-604, 2001.
36. W. Y. Loh, "Regression trees with unbiased variable selection and interaction detection", *Statistica Sinica*, vol. 12, pp. 361-386, 2002.
37. P. Chaudhari, M. C. Huang, W. Y. Loh, R. Yao, "Piecewise-polynomial regression trees", *Statistica Sinica*, vol. 4, pp. 143-167, 1994.
38. P. Chaudhari, W. Y. Loh, "Quantile regression trees", *Technical Report 994*, Department of Statistics. University of Wisconsin, 1998.
39. R. Kohavi, "Bottom-up induction of oblivious read-once decision graphs", *in proceedings of the European conference on Machine Learning*, ECML-94, pp. 154-169, 1994.

40. R. Kohavi, "Bottom-up induction of oblivious read-once decision graphs: strengths and limitations", *in proceedings of the Twelfth National Conference on Artificial intelligence*. AAAI'94, pp. 613-618, 1994.
41. R. Kohavi, C. H. Li, "Oblivious decision trees, graphs, and top-down pruning", *in proceedings of the Fourteenth International joint conference on Artificial intelligence (IJCAI-95)*, vol. 2, pp. 1071-1077, 1995.
42. P. Langley, S. Sage, "Oblivious decision trees and abstract cases", *AAAI Technical Report WS-94-01*, The Association for the Advancement of Artificial Intelligence, pp. 113-117, 1994.
43. H. Almuallim, T. G. Dietterich, "Learning with many irrelevant features*", in Proceedings of the Ninth National conference on Artificial intelligence*, pp. 547-552, 1991.
44. J. C. Schlimmer, "Efficiently inducing determinants: a complete and systematic search algorithm that uses optimal pruning", *in Proceedings of the Tenth International Conference on Machine Learning*, pp. 284-290, 1993.
45. J. C. Schlimmer, D. Fisher, "A case study of incremental concept induction", *in Proceedings of the Fifth AAAI National Conference on Artificial Intelligence*, pp. 496-501, 1986.
46. P. E. Utgoff, "Incremental induction of decision trees", *Machine Learning*, vol. 4, no. 2, pp. 161-186, 1989.
47. P. E. Utgoff, N. C. Berkman, J.A. Clouse, "Decision tree induction based on efficient tree restructuring", *Machine Learning*, vol. 29, no. 1, pp. 5-44, 1997.
48. P. Domingos, G. Hulten, "Mining high-speed data streams*", in proceedings of the Sixth ACM SIGKDD International Conference on Knowledge discovery and data mining*, pp. 71-80, 2000.
49. G. Hulten, L. Spencer, P. Domingos, "Mining Time-changing data streams", *in proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 97-106, 2001.
50. C. Lian, Y. Zang, Q. Song, "Decision tree for dynamic and uncertain data streams", *in JMLR: Workshop and Conference Proceedings 13, Second Asian conference on Machine Learning (ACML2010)*, pp. 209-224, 2010.
51. J. Basak, "Online adaptive decision trees", *Neural Computation*, vol. 16, no. 9, pp. 1959-1981, 2004.
52. B. Panda, J. S. Herbach, S. Basu, R. J. Bayardo, "PLANET: massively parallel learning of tree ensembles with MapReduce", *Proceedings of the VLDB Endowment*, vol. 2, no. 2, pp. 1426-1437, 2009.
53. R. Jin, G. Agrawal, "Communication and memory efficient parallel decision tree construction", *in Proceedings of the 2003 SIAM International Conference on Data Mining*, pp. 119-129, 2003.
54. K. Alsabti, S. Ranka, V. Singh, "CLOUDS: a decision tree classifier for large datasets", *in Proceeding KDD'98 Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*, pp. 2-8, 1998.
55. M. K. Sreenivas, K. Alsabti, S. Ranka, "Parallel out-of-core divide-and-conquer techniques with application to classification trees", *in Proceedings 13th IEEE International Parallel Processing Symposium and 10th Symposium on Parallel and Distributed Processing*, pp. 555-562, 1999.
56. J. Gehrke, V. Ganti, R. Ramakrishnan, W. Y. Loh, "BOAT- optimistic decision tree construction*", in Proceedings of the 1999 ACM SIGMOD International Conference on Management of data*, pp. 169-180, 1999.
57. Y. Ben-Haim, E. Tom-Tov, "A streaming parallel decision tree algorithm", *Journal of Machine Learning Research*, vol. 11, pp. 849-872, 2010.
58. C. A. Rosset, "A review of online decision tree learning algorithm", 2015, Available online at http://corbyrosset.com/files/OnlineDecisionTreeReview.pdf. Accessed 28 September 2018.
59. P. E. Utgoff, C. E. Brodley, "Linear machine decision trees", *Technical Report 91-10*, Department of Computer SCIENCE, University of Massachusettes, Amhersts, Massachusettes. 1991
60. N. J. Nilsson, "Introduction to machine learning", Early draft of the book. 1998. (pg. 50) Available online https://ai.stanford.edu/~nilsson/MLBOOK.pdf. Accessed on 2 March 2018.
61. D. Heath, S. Kasif, S. Salzberg, "Induction of oblique decision trees", *in Proceedings of the 13th International joint conference on artificial intelligence*, pp. 1002-1007, 1993.
62. A. K. Murthy, S. Kasif, S. Salzberg, "A system for induction of oblique decision trees", *Journal of Artificial Intelligence Research*, vol. 2, pp. 1-32, 1994.
63. E. Cantu-Paz, C. Kamath, "Inducing oblique decision trees with evolutionary algorithms", *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 1, pp. 54-68, 2003.
64. A. Lopez-Chau, J. Cervantes, L. Lopez-Garcia, F. G. Lamont, "Fisher's decision tree", *Expert Systems with Applications*, vol. 40, no. 16, pp. 6283-6291, 2013.
65. N. Manwani, P. S. Sastry, "Geometric Decision tree", *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 42, no. 1, pp. 181-192, 2010.
66. D. C. Wickramarachchi, B. L. Robertson, M. Reale, C. J. Price, J. Brown, "HHCART: an oblique decision tree", Computational *Statistics & Data Analysis*, vol. 96, pp. 12-23, 2016.
67. E. Frank, M. A. Hall, I. H. Witten, "The WEKA Workbench", *Online appendix for 'Data Mining: Practical machine learning tools and techniques'*, Morgan Kaufmann. 2016.
68. Proc DTREE. SAS/OR® 14.1 User's Guide: Project Management . Cary, NC: SAS Institute Inc.
69. Proc HPSPLIT. SAS Institute Inc. 2015. SAS/STAT® 14.1 User's Guide. Cary, NC: SAS Institute Inc.
70. The ARBORATUM Procedure. http://support.sas.com/documentation/onlinedoc/miner/em43/allproc.pdf Accessed 20 September 2018.
71. Decision tree models. https://www.ibm.com/support/knowledgecenter/en/SS3RA7_15.0.0/com.ibm.spss.modeler.help/nodes_treebuilding.htm. Accessed 20 September 2018.
72. A. Janosi, W. Steinbrunn, M. Pfiesterer, R. Detrano, "Heart Disease data set", UCI Machine learning repository. Accessed 20 September 2018.

Authors Profile

Pravin S. Game, is a Ph.D. scholar in Computer Engineering at Shri JJT University, Jhunjhunu, Rajasthan. He received his Master of Engineering in Computer Engineering from Savitribai Phule Pune University and Bachelor of Engineering in Computer Science & Engineering from Sant Gadge Baba Amravati University. Currently, he is working at Pune Institute of Computer Technology, Pune. His research interests include data mining, big data analysis, machine learning.
Email:pravinsgame@gmail.com.

Dr. Vinod Vaze, is Ph.D. in Computer Engineering. He is B. Tech. from I.I.T., Kanpur, and has also earned PGDFM, Diploma in Cyber Law. He is currently working in Department of Computer Science and Engineering at Shri JJT University, Jhunjhunu, Rajashtan. His research interest includes machine learning, and cyber security.

Dr. Emmanuel M, . is Ph.D. in Computer Science and Engineering. He is M. Tech. and B. Tech. in Computer Science and Engineering. He is currently working at Pune Institute of Computer Technology, Pune. He is leading the Big Data research group at PICT. His research interest includes big data, business intelligence, medical image processing and machine learning.