International Journal of Applied Information Systems (IJAIS) – ISSN : 2249-0868
Foundation of Computer Science FCS, New York, USA
Volume 3 – No.5, July 2012 – www.ijais.org

Issues in Optimization of Decision Tree Learning: A Survey

Dipak V. Patil
Department of Computer Engineering
Sandip Institute of Technology and Research Centre,
Nashik, M.S., India.

R. S. Bichkar
Department of Electronics & Tele. Engg.
G.H. Raisoni College of Engineering and
Management, Pune, M.S., India.

ABSTRACT

Decision tree induction is a simple but powerful learning and classification model. Decision tree learning offers tools for discovery of relationships, patterns and knowledge from data in databases. The volume of data in databases is growing to quite large sizes, both in the number of attributes and instances. Decision tree learning from a very large set of records in a database is quite complex task and is usually a very slow process, which is often beyond the capabilities of existing computers. There are various issues and problems related to decision trees. To handle these issues various approaches have been proposed in the past by different researchers. This paper is an attempt to summarize the proposed approaches, tools etc. for decision tree learning with emphasis on optimization of constructed trees and handling large datasets.

Keywords*:* Decision Tree, Optimization.

1. INTRODUCTION

Decision tree induction is a simple yet powerful learning and classification model. Decision tree learning offers tools for discovery of relationships, patterns and knowledge from data in databases. The volume of data in databases is growing to quite large sizes, both in the number of attributes and instances. Decision tree learning using a very large set of records in a database is quite complex task and is usually a very slow process, which is often beyond the capabilities of existing computers. There are various issues related to decision trees and training data sets. To handle these issues various approaches have been proposed in the past by different researchers. This paper summarizes these proposed approaches to handle various issues related to decision tree learning and issues related to handling problems with data. The emphasis is given on issues which help to optimise the process of decision tree learning.

A decision tree is a classifier in the form of tree structure that contains decision nodes and leaves**.** It assigns a class value to an instance. In tree construction process, partitioning of attributes is according to splitting criteria that implements better use of available attributes and also implies computational efficiency in classification. The tree construction takes polynomial time concerning the number of attributes and inputs, as no backtracking is required [1]-[7].

Some of the important approaches and issues are been introduced in this paper. The issues are:

1. Decision tree learning algorithm and various splitting criteria such as gain, gini index, twoing rule etc.
2. Pruning techniques-these includes the techniques for pruning overfitted decision tree technique. Some prominent methods includes error based pruning, cost complexity pruning and reduced error pruning.
3. Decision tree learners- decision tree learning algorithms such as CHAID, ID3, C4.5 and CART.
4. Data preprocessing- Data preprocessing includes t echniques for feature subset selection, data sampling, outlier detection and handling missing data. This topic is generalized and is not restricted to techniques for decision tree.
5. Soft computing approach for decision tree learning & optimization these include use of neural networks, Evolutionary techniques and Fuzzy logic.
6. Handling large data set approaches such as parallel, distributed, scalable and Meta decision tree.
7. Surveys on decision tree learning.
8. Decision tree learning softwares available and some of the commonly used benchmark datasets.
9. Other issues like incremental induction of decision tree and oblique decision trees.
10. Applications of decision trees in various areas.

Decision tree algorithms construct trees by recursively partitioning a training set. A training set consists of set of attributes and a class label. An attribute can have real, Boolean or ordinal values. A decision node states a test to be carried on a particular attribute value of an instance. A branch is present for each probable output of the test. Thus, a tree is traversed from the root to a leaf of the decision tree to identify the class of the instance. The specified class at the leaf is the classification by the decision tree. The classification accuracy, defined as the percentage of correctly classified instances in the test data, specifies the performance of decision tree. The generalized decision tree algorithm is explained here.

1.1 The Tree Construction Algorithm

The tree construction algorithms use a divide and conquer approach to construct a decision tree. It evolves a decision tree for a given training set T consisting of set of training instances. An instance denotes values for a set of attributes and a class. Let the classes be denoted by $\{C_1, C_2, \dots, C_n\}$. Initially, the class frequency is computed for instances in training set T. If all instances belong to same class, node K with that class is constructed. However, if set T contains instances belonging to more than one class, the test for selecting attribute for splitting is executed and the attribute satisfying splitting criteria is chosen for the test at the node. The training set *T* is then partitioned into *k* exclusive subsets $\{T_1, T_2, \ldots, T_k\}$ on the basis of this test and the algorithm is recursively applied on each nonempty partition. The algorithm for construction of a decision tree is given below.

```
Construct (T)
1. Calculate freq ($C_i$, $T$).
2. If (all instances belong to same class), return leaf.
3. For every attribute A test for splitting criteria.
   Attribute satisfying test is test node K.
4. Recur Construct ($T_i$) on each partition $T_i$.
   Add those nodes as children of node K.
5. Stop.
```

2. Selecting the Best Attribute for Splitting

The selection of test attribute at each node in the decision tree is an important process. Various approaches have been proposed to select the best possible attribute. The approaches are categorized by Ben-Bassat [6] as use of information theory, distance measure and dependence measure. Some of the approaches are discussed below.

2.1 Using Information Theory

Last and Maimon [7] expressed that objective of process of data classification is to maximize the information gain as it leads to increase in classification accuracy. Quinlan [2] used information gain and gain ratio in decision tree algorithm. The gain is defined as the information obtained from a message based on its probability *P*. For any set of instances *T*, the probability that an instance belongs to class C_i is given as

$$P = \frac{\text{freq}\ (C_i, T)}{|T|} \qquad (1)$$

Where $|T|$ is number of instances in set T and freq (C_i, T) denotes the number of instances in T that belong to class C_i.. Now, the average information contained in set *T* regarding class association of instances, called entropy of set *T*, and is calculated in bits as

$$\text{info}(T) = \sum_{i=1}^{k} P_i \times \log_2 (P_i) \text{ bits.} \qquad (2)$$

Where *k* is the number of classes in set *T*. The test *X* performed at a node on the preferred attribute provides subsets $T_1, T_2, \ldots, T_k$. The information by this partitioning process is calculated as the sum over these subsets, is given as

$$\text{info}_x(T) = \sum_{i=1}^{k} \frac{|T_i|}{|T|} \times \text{info}(T_i) \qquad (3)$$

The reduction in entropy due to partioning of *T* with test *X* on the preferred attribute, denoted as Gain (*X*), is calculated as

$$\text{Gain}\ (X) = \text{info}(T) - \text{info}_x(T) \qquad (4)$$

The attribute which provides maximum information gain is selected. The problem with above approach for selection of test attribute at a node is that it is biased towards attributes with many values as compared to attributes with less values and it leads to large decision trees that poorly generalize the problem. This problem can be eliminated using normalization of gain criterion and use of gain ratio. The gain ratio calculates ratio of information generated by partioning *T* and is expressed as

$$\text{gain ratio (X)} = \text{gain (X)} / \text{split info(X)} \qquad (5)$$

The split info(*X*) calculates information gained by splitting training set *T* into *k* subsets on test *X*.

$$\textit{split}\ \text{info(X)} = -\sum_{i=1}^{k} \frac{|T_i|}{|T|} \times \log_2\left(\frac{|T_i|}{|T|}\right) \qquad (6)$$

The attribute on which test obtains maximum gain ratio is selected. This approach has problem that it tends to favour attributes for which split info(*X*) is very small. Another problem is that gain ratio can be calculated only when the split info(*X*) is nonzero. To overcome this problem, Quinlan suggested computing information gain over all attributes and considering attributes with information gain which is at least as large as average of information gain over all attributes. The use of gain ratio provides better accuracy and complexity of classifier.

Jun et al. [8] suggested a modification in above entropy calculation where the base of the logarithm is the number of successors to the node and have shown that this approach can handle huge amount of data efficiently.

2.2 Using Distance Measure

A classification and regression tree uses *gini index* as impurity measure for selecting attribute that is based on distance measure. These attribute evaluation criteria computes separability, deviation or discrimination between classes [9]. For a data set *T*, *gini index* is defined as

$$gini(T) = 1 - \sum_{i=1}^{n} p_i^2 \qquad (7)$$

Where p_i indicates the relative frequency of class *i* in the data set *T*. The attribute with the largest reduction in impurity is used for splitting the node's records. After splitting *T* into two subsets T_1 and *T2* with sizes *N1* and *N2* respectively then the *gini index* of the split data is defined as

$$gini(T)_{split} = \frac{N1}{N} gini(T_1) \frac{N2}{N} gini(T_2) \qquad (8)$$

Breiman *et al.* [9] identified that the *gini index* has a problem in handling large number of classes. In such a case, binary criterion called *twoing index* is used which is based on dividing the multiple classes into two super classes and then calculating the best split on the attribute based on these super classes. Murthy et al. [10] explained this twoing rule as follows. In the beginning the set is the complete training set *T*, is divided into two non-overlapping subsets, T_L and T_R by hyperplane *H*. The impurity measure at the start checks if T_L and T_R are homogeneous and belongs to the same category and in that case return minimum zero impurity. The value to be computed is defined as

$$Twoingvalue = \frac{|T_L|.|T_R|}{n^2}\left(\sum_{i=1}^{k}\left|\frac{L_i}{|T_L|}-\frac{R_i}{|T_R|}\right|\right)^2 \qquad (9)$$

Where $|T_L|$ and $|T_R|$ represents the number of instances on the left and right of a split at node T, and the number of instances at node *T* are represented by *n*. The number of instances in category *i* on the left and right of the split are represented by L_i and R_i respectively.

Mantras [11] introduced distance based attribute selection measure. The attribute selected with this criterion is in partition, a partition which is closest to the correct partition of the subset of training set related to the node. Chandra et al. [12] proposed a novel node splitting criteria called as distinct class based splitting measure (DCSM) for decision tree induction. It is motivated by concept of gini index. The measure gives importance to the number of distinct classes in a partition. The DCSM criterion is combination of the product of two terms. The first term handles the number of diverse classes in every child partition. With increase in number of different classes in a partition the first term increases. Consequently these purer partitions are favored. The second term decreases when there are more instances of a particular class as against the total number of instances in that partition. As a result amalgamation favors purer partitions.

Rounds [13] presented attribute selection criteria based on the Kolmogorov-Smirnov distance that keeps an optimal classification decision at each node. Utgoff and Clouse [14] used the same Kolmogorov-Smirnov distance measure with improvements to take care of attribute with multiclass and missing attribute values. Martin [15] has done empirical comparative analysis of splitting methods like distance, orthogonality, a Beta function and two chi-squared tests. Buntine and Niblett [16] recommended alternative investigational methods and presented additional results for splitting rules for decision tree induction.

3. Decision Tree Pruning

Decision trees are often large and complex and as a result they may become inaccurate and incomprehensible. The causes of overfitting in decision trees are the noisy data, unavailability of training samples for one or more classes and insufficient training instances to represent the target function. Decision tree pruning removes one or more subtrees from a decision tree. It makes overfitting trees more accurate in classifying unseen data.

Various methods have been proposed for decision tree pruning. These methods selectively replace a subtree with a leaf, if it does not reduce classification accuracy over pruning data set. Pruning may increase the number of classification errors on the training set but it improves classification accuracy on unseen data.

Pruning techniques can be divided into two groups. The techniques in first group approximately compute the probability of misclassification of a subtree and then make pruning decision using an independent test set called pruning data set. In second group, iterative grow and prune method is used during the construction of a tree. Some important pruning methods are reviewed here briefly.

3.1 Cost Complexity Pruning

Cost complexity pruning [9] is used in the CART system. Starting with initial unpruned tree that is constructed from complete training set, this algorithm constructs a chain of progressively smaller pruned trees by replacing one or more subtress best possible leaves. The method prunes those subtrees that give the lowest increase in error for the training data. The cost complexity of a tree is defined as ratio of number of correctly classified instances to misclassified instances in training data plus number of leaves in that tree multiplied by some parameter α. [17] - [19].

3.2 Reduced-Error Pruning

The reduced error pruning proposed by Quinlan [19] is a bottom-up approach in which the non-leaf subtrees are replaced with best possible leaf nodes if these replacements reduce the classification error on the pruning data set.The process continues towards the root node until the pruning decreases error. The process assures smallest and most accurate decision trees with respect to the test data [17]-[20].

3.3 Critical Value Pruning

Mingers [20] proposed critical value pruning, which uses the information gathered during tree construction. It sets a threshold called a critical value to select a node for pruning. Various measures such as gain, info gain etc. can be used to select the best attribute at the test node. If the value of selection criterion is smaller than this threshold value the subtree is replaced with a best possible leaf. However, if the subtree contains at least one node having value greater than the threshold, the subtree cannot be pruned [17], [18].

3.4 Minimum Error Pruning

Niblett and Bratko [17] proposed Minimum-error pruning, which is a bottom-up approach. To get error estimates of a subtree to be pruned, the errors for its children are estimated. The dynamic error of the node is calculated as weighted sum of static errors of its children. If dynamic error of *t* is less than its static error, *t* will be pruned and will be replaced with best possible leaf.

3.5 Pessimistic Error Pruning

Pessimistic error pruning, a top down approach proposed by Quinlan [19] uses error rate estimates to make decisions concerning pruning the subtrees similar to cost complexity pruning. It calculates classification errors on training data and does not require separate pruning set. Since the classification errors estimated from training set cannot provide best pruning results for unseen data, this pruning technique assumes that each leaf classifies a certain fraction of instances with error. To reflect these errors, it adds a continuity correction for binomial distribution to the derived training error of a subtree. However, as the corrected misclassification estimation by a subtree is expected to be optimistic, the algorithm calculates standard error. Quinlan recommends pruning a subtree if its corrected estimate of error is lower than that for the node by at least one standard error [18], [20].

3.6 Error-Based Pruning

Error-based pruning is the default pruning method for the well-known C4.5 decision tree algorithm [2]. Instead of using a pruning set it uses error estimates. The method assumes that the errors are binomially distributed and calculates error estimates from the training data. The number of instances covered by the leaf of the tree are used to estimate errors.

A bottom-up approach [18] is used for error-based pruning. If the number of predicted errors for the leaf is not greater than the sum of the predicted errors for the leaf nodes of that subtree then subtree is replaced with that leaf.

3.7 Minimum description length pruning

Mehata et al. [21] and Quinlan and Rivest [22] utilized MDL principle for decision tree pruning. The principle of minimum description length used here states that a classifier that compresses the data is a preferable inducer. The MDL pruning method selects decision tree with less number of bits required to represent it. The size of the decision tree is measured as number of bits required for encoding the decision tree. The method searches for decision tree that maximally compresses the data [18].

3.8 Optimal Pruning

Optimal pruning algorithm constructs smaller pruned trees with maximum classification accuracy on training data. Breiman et al. [9] first suggested a dynamic programming solution for optimal pruning algorithm. Bohanec and Bratko [23] introduced an optimal pruning algorithm called OPT which gives better solution. Almuallim [24] proposed an enhancement to OPT called OPT-2. It is also based on dynamic programming and has flexibility in various aspects and is easy to implement.

3.9 Improvements to Pruning Algorithms

Matti Kääriäinen [25] analyzed reduced error pruning and proposed a new method for obtaining generalization of error bounds for pruning the decision trees. Error-based pruning has been blamed for the general effect of under-pruning. Hall et al. [26] proved that if the certainty factor value CF is appropriately set for the data set, error-based pruning constructs trees that are essentially steady in size, in spite of the amount of training data. The CF calculates the upper limit of the probability of an error at a leaf. Oates and Jenson [27] presented improvements to reduced error pruning to handle problems with large data sets. Macek and Lhotsk [28] presented a technique for pruning of decision trees based on the complexity measure of a tree and its error rate. The technique utilizes the Gaussian complexity averages of a decision tree to compute the error rate of classification. Frank [29] enhanced performance of standard decision tree pruning algorithm. The performance is enhanced with statistical significance of observations. Bradford et al. [30] proposed pruning decision trees with misclassification cost with respect to loss. Scott [31] proposed algorithms for size-based penalties and subadditive penalties.

Bradley and Lovell [32] proposed a pruning technique that is sensitive to the relative costs of data misclassification. They implemented two cost-sensitive pruning algorithms, by extending pessimistic error pruning and minimum error pruning technique. Cai1 et al. [33] proposed cost-sensitive decision tree pruning CC4.5 to deal with misclassification cost in the decision tree. It provides three cost-sensitive pruning methods to handle with misclassification cost in the decision tree. Mansour [34] proposed pessimistic decision tree pruning based on tree size. A graphical frontier-based pruning (FBP) algorithm is proposed by Huo et al. [35] which provides a full spectrum of information while pruning the tree. The FBP algorithm starts from leaf nodes and proceeds towards root node with local greedy approach. The authors further proposed combination of FBP and cross validation method.

In decision tree learning pre-pruning handles noise and post-pruning handles the problem of overfitting. Faurnkranz [36] proposed two algorithms to combine pruning and post pruning operations. A method for pruning of oblique decision trees was proposed by Shah and Sastry [37].

3.10 Comparison of Pruning Methods

The empirical comparative analysis is one of the important methods to compare the performance of various available algorithms. Quinlan [19] examined and empirically compared tree cost complexity pruning, reduced error pruning and pessimistic pruning on some data sets. These methods have demonstrated significant improvement in terms of size of the tree. Cost complexity pruning tends to produce smaller trees than reduced error pruning or pessimistic error pruning where as in case of classification accuracy, reduced error pruning is somewhat superior to Cost complexity pruning.

Floriana et al. [18] presented comparative analysis six well-known pruning methods. Each method has been critically reviewed and its performance has been tested. The paper provides study of theoretical foundations, computational complexity and strengths and weaknesses of the pruning methods. According to this analysis, reduced-error pruning outperforms other methods. In addition, MEP, CVP and EBP tend to under prune whereas reduced-error pruning tends to over prune.

Similarly, Mingers [19] analyzed five pruning methods with four different splitting criteria. The author has provided the analysis based on size and accuracy of the tree. This work showed that minimum-error pruning is extremely sensitive to the number of classes in the data and is the least accurate method. Pessimistic error pruning is bad on certain datasets and needs to be handled with care. Critical value, cost complexity, and reduced-error pruning methods produced trees with low error rates on all the data sets with consistency. He further clarified that there is no evidence of relation between splitting criteria and pruning method. Windeatt [17] presented empirical comparison of pruning methods for ensemble classifiers. It has been proved that error based pruning performs best for ensemble classifiers. From above studies we can conclude that reduced error pruning and cost complexity pruning methods are the promising pruning methods as compared to other available methods.

4. Decision Tree Learners

Researchers have developed various decision tree algorithms over a period of time with enhancement in performance and ability to handle various types of data. Some important algorithms are discussed below.

CHAID: CHAID (CHi-squared Automatic Interaction Detector) is an initial decision tree learning algorithm, which is an extension of the AID (Automatic Interaction Detector) and THAID (Theta Automatic Interaction Detector) procedures. It works on principal of adjusted significance testing. It was developed by Kass [38] in 1980. CHAID is easy to interpret and can be used for classification and detection of interaction between variables. After detection of interaction between variables it selects the best attribute for splitting the node, such that each child node is made of a collection of homogeneous values of the selected attribute. The method can handle missing values. It does not imply any pruning method.

ID3: ID3 (Iterative Dichotomiser 3) decision tree algorithm is developed by Quinlan [39]. It is based on Occam’s razor, which states that simplest hypothesis should be adopted. Here Occam’s razor is incorporated through use of information entropy. ID3 uses information gain as splitting criteria. Information gain decides how effectively the attribute separates training instances according to their class. ID3 does not use pruning method. It cannot handle numeric attributes and missing attribute values. When training data contains noise the performance of the algorithm is degraded.

C4.5: The C4.5 algorithm [8] is improvement over ID3 algorithm. The algorithm uses information gain as splitting criteria. It can accept data with categorical or numerical values. To handle continuous values it generates threshold and then divides attributes with values above the threshold and values equal to or below the threshold. The default pruning method is error-based pruning. As missing attribute values are not utilized in gain calculations the algorithm can easily handle missing values.

CART: Classification and regression tree (CART) proposed by Breiman *et al.* [9] constructs binary trees. The word binary implies that a node in a decision tree can only be split into two groups. CART uses gini index as impurity measure for selecting attribute. The attribute with the largest reduction in impurity is used for splitting the node's records. It can accept data with categorical or numerical values and also handle missing attribute values. It uses cost-complexity pruning. It can also generate regression trees.

5. Data Pre-Processing

The knowledge discovery process consists of an iterative sequence of subtasks such as selection of data subset handling noise and missing data etc. Data pre-processing techniques used in decision trees are discussed here.

5.1 Feature Subset Selection

The real-world applications provide us numerous attributes that can be used for learning. When given data set contains large number of attributes the classification performance of inductive method may degrade. The solution is to use a quality subset of those attributes. Feature selection improves performance of learning algorithms by finding a minimal subset of relevant features. Feature selection removes irrelevant, noisy and repeated features and keeps the most relevant features.

Ron Kohavi [40] proposed the use of a search with probabilistic estimates for probing a space. The method results in accurate and comprehensible trees. Caruana and Freitag [41] proposed a caching system with hill climbing in attribute space for feature subset selection and found that hill climbing in attribute space can get significant improvement in classification performance. Mark Last et al. [42] proposed an algorithm that computes fuzzy information gain as quality measure. The resulting tree size is reduced. Wu and Flach [43] proposed merging of heuristic measures and exhaustive search method to get optimal subset. Goodman-Kruskal measure and Fisher’s exact test are used to rank the feature.

Huan et al. [44] proposed a monotonic measure, which is accurate and fast. The proposed method reduces error rate. Grabczewski and Jankowski [45] proposed two feature selection algorithms that are based on separability of spilt value criteria and verified that they are good alternatives to the available most popular methods with respect to classification accuracy. Bins and Draper [46] proposed three-stage algorithm for large data sets. The algorithm is combination of the relief algorithm to remove irrelevance, K-means to remove redundancy and a standard combinatorial feature selection algorithm.

Several authors have proposed evolutionary computation based methods for feature selection. Guerra-Salcedo et al. [47] proposed hybrid genetic feature selection approach that is fast and accurate. Landry et al. [48] proposed a feature selection technique based on the genetic programming. Bala et al. proposed [49] use of genetic algorithms for evaluations of features.

Filter and wrapper approaches are also used for feature subset selection. Filters remove irrelevant features from data set. Filters work independently of any induction and it takes place before induction process. The filter approach does not consider the effect of subset selection on performance of induction algorithm.

The Wrapper approach uses a statistical re-sampling technique such as cross validation along with the induction algorithm. The induction algorithm with some objective function is used to evaluate the selected feature subset. Legrand and Nicolas [50] proposed a hybrid technique that combines filter and wrapper approaches with principle of preferences aggregation. Hall and Smith [51] proposed a correlation based filter approach to select feature subset. Duch et al. [52] proposed inexpensive filters based on information theory. Hall [53] proposed faster and accurate filter algorithm useful for continuous and discrete domains. Yuan et al. [54] proposed a two-stage feature selection algorithm of filter and wrapper approach to get benefit of both approaches. Initially the filter approach eliminates irrelevant features and then the wrapper approach eliminates redundant features. Lanzi [55] proposed genetic algorithm with filter that is faster than standard genetic algorithm for feature selection.

5.2 Outlier Detection

The problem of outlier detection and noise elimination is an important issue in data analysis. The removal of outliers improves data quality and hence classification performance. Several researchers have proposed various approaches for data cleaning. These include use of MDL principle, neural networks, filters, Occam's razor and some other methods. The approaches are discussed below.

John George [56] proposed a method that removes a misclassified training instance from training data and rebuilds tree on filtered data, the process is repeated till all such instances are removed from training data. Misclassified instances are identified using tree classifier as a filter. The classifier built on clean data improves prediction accuracy.

Arning et al. [57] proposed framework for the problem of outlier detection. Similar to human beings, it observes all instances for similarity with data sets and it treats dissimilar data set as an exception. A dissimilarity function is used to find out an outlier.

Guyon et al [58] proposed training of convolutional neural networks with local connections and shared weights. The neural network is trained with minimizing mean-square-error cost function with backpropogation algorithm. Unclean data is applied as input and several increasing strict levels of cleaning are forced. The classification error is used as an objective function.

Gamberger and Lavrac [59] proposed conditions for Occam's razor applicability in noise elimination. The Occam's principle states that when there are many hypotheses, one should choose simplest hypothesis that is correct for all the training instances or maximum training instances. This hypothesis is expected to capture the information inbuilt in the problem and provide accurate classification accuracy on unseen data.

Knorr and Ng [60] proposed unified outlier detection system. Subsequently, they [61] proposed and analyzed some algorithms for detecting distance-based outliers. Tax and Duin [62] proposed outlier detection that is based on the instability of the output of simple classifiers on new objects.

Broadly and Friedl [63] introduced a method for detecting mislabeled instances. The approach is to employ a set of learning algorithms to build classifiers that act as a filter for the training data. The method is based on the technique of removing outliers in regression analysis [64]. An outlier in this case is an instance that comes from a different probability distribution.

Gamberger et al. [65] proposed saturation filter. It is based on principle that detection and removal of noisy instances from training data to induce less complex and more accurate hypothesis. The removal of noisy instances from training data reduces the complexity and the Less Complex Correct Hypothesis (CLCH) value. The saturation filter checks the saturation of training data with use of CLCH value. A source of all possible correct instances in a given problem domain is called target concept. A good representation of target concept in the inducted hypothesis is called target theory. The saturated training data set can be employed for induction of stable target theory.

Schwarm and Wolfman [66] proposed Bayesian methods for data cleaning; the method detects errors and corrects errors using Bayesian methods. The Bayesian methods utilizes dependencies between attributes in participated manner and uses expert knowledge of the relationships between the attributes. Ramaswamy et al. [67] proposed algorithm for distance-based outliers that is based on the distance of a point from its nearest neighbor. The solution ranks each point based on its distance to its nearest neighbor. The higher points in this ranking indicate outliers.

Raman and Hellerstein [68] proposed an interactive framework for data cleaning that integrates transformation and discrepancy detection. This framework progressively constructs transformations by adding or undoing transforms, in an instinctive, graphical manner using a spreadsheet-like interface. The effect of a transform on instances is available immediately on screen. In the background, the system continues to infer the structure of the data in terms of user-defined domains and uses appropriate algorithms to check it for outliers. The proposed frame structure progressively constructs a transformation as outliers are found, and cleans the data without scripting complex programs or enduring long delays.

Kubika and Moore [69] presented system for learning explicit noise. The system detects corrupted fields and uses non-corrupted fields for consequent modeling and analysis. Verbaeten and Assche [70] proposed three ensemble based methods for noise elimination in classification problems. The first one is base classification algorithm in which ILP extension of C4.5 is used. This extension uses logical queries as test values instead of attribute values at test nodes. The second filter technique proposed is voting filter removes outlier if all or majority of classifiers misclassify the instance. The last technique is boosting filters in this method Adaboost is used and after n number of rounds instances with highest weighs are removed. Loureiro et al [71] proposed a method that applies hierarchical clustering methods for outlier detection. Xiong et al. [72] proposed a hyperclique-based noise removal system to provide superior quality association patterns. The hyperclique pattern contains items those are strongly correlated to each other. The existence of an item in one matter strongly indicates the existence of every other item that belongs to the same hyperclique pattern. The h-confidence threshold designates the strength of this association. The higher the threshold, the stronger is the relationship. The system discovers all hyperclique patterns for a given h-confidence threshold and removes any objects that are not belonging to any hyperclique pattern.

Seung Kim et al. proposed [73] fast outlier detection for very large log data S. Hido et al. [74] proposed statistical outlier detection using density ratio estimation.

5.3 Data Sampling

Sampling is the process of taking a subset of instances that represents the entire population. The representativeness is the primary concern in statistical sampling. Sampling is done since it is impossible to test every single individual in the population. It is also desirable to save time, resources and effort while conducting the research. The sample must have sufficient size to justify statistical analysis. George John and Pat Langley [75] experimented on static and dynamic sampling and found that dynamic sampling is robust as compared to static sampling. Jenson and Oates [5] experimented on data sampling and proved that as size of the training dataset increases, size of tree also increases where as classification accuracy does not increase significantly. Foster et al. [76] concluded that progressive sampling can be remarkably efficient. Patil and Bichkar [77] proposed use of evolutionary decision tree with sampled data to optimise the problem and found that the proposed method builds trees that are accurate and relatively smaller in size.

5.4 Handling Missing Attribute Values

Missing attribute values is one of the important and common problems in the real world data sets. While collecting data some attribute values from a tuple are lost. It creates problem for training as well as testing, because it reduces classification accuracy. Several researchers have addressed the problem of handing missing attribute values. Little and Rubin [78] divided the methods for handling missing data into three categories; the categories are ignoring and discarding data, parameter estimation, and imputation. Imputation is procedure of substituting missing values of attributes with some plausible values. Imputation is further divided as case substitution, mean or mode imputation, hot and cold deck and prediction model. Batista [79] experimented k-nearest neighbour imputation and found that it performs better than mean or mode imputation method of C4.5 algorithm. Friedman et al. [80] suggested ignoring every tuple with missing attribute values from training instances. Authors found that it may result in loss of bias information due to ignoring.

Quinlan experimented on problem of missing attribute values [81] and concluded that ignoring cases with missing values hampers the classification accuracy and depends on attribute to attribute. Quinlan substituted missing values with most common test outcomes and found that it performs well in some cases but poorly in others.

Kuligowski and Barros [82] proposed a use of a back propagation neural network for estimation of missing data by using concurrent rainfall data from neighboring gauges. Brockmeier et al [83] experimented empirical comparative analysis of deletion and imputation Techniques.

Abebe et al. [84] proposed a use of a fuzzy-rule-based model for substitution in missing rainfall data using data from neighboring stations. The authors have provided empirical comparative analysis of results using the fuzzy-rule-based model and results using an ANN model and a traditional statistical model. The fuzzy-rule-based model performs slightly better.

Sinharay et al. [85] experimented on the use of multiple imputations for the analysis of missing data. Khalil et al. [86] proposed cyclic federation of data intended for budding ANN models to estimate missing values in monthly surplus datasets. Bhattacharya et al. [87] used ANN models to substitute the missing values of wave data. Fessant and Midenet [88] proposed use of a self-organizing map (SOM) for imputation of data along with the multilayer perceptron (MLP) and hot deck methods.

Musil et al. [89] provided empirical comparative analysis on list wise deletion, mean substitution, simple regression, regression with an error term and the EM algorithm. Junninen et al. [90] experimented on univariate linear, spline and nearest-neighbor interpolation algorithm, multivariate regularized expectation–maximization algorithm, nearest-neighbor, self-organizing map, multilayer perceptron (MLP) as well as hybrid methods.

M. Subasi, et al. [91] proposed new imputation method for incomplete binary data. Amman Mohammad Kalteh & Peder Hjorth [92] experimented on imputation of missing values with self organizing map, multilayer perceptron, multivariate nearest neighbor, regularized expectation maximization algorithm and multiple imputation for precipitation runoff process data set. Rhian M. et al. [93] proposed a method for increasing the robustness of multiple imputations.

Patil and Bichkar [94] proposed multiple imputation of missing data with genetic algorithm based techniques. Authors proposed to use the domain values of an attribute as pool of solution for categorical data. The method improves the classification accuracy of training data.

6. Class Distribution

The classifier performance is affected by varying class distribution of the training instances. The experiments by Gary Mitchell Weiss [95] specify that the naturally happening class distribution is not all the time best for learning. A balanced class distribution should be preferred to make a robust classifier. To reduce learning cost it is essential to control the quantity of training data used for learning. Gary Mitchell Weiss proposed a budget-sensitive progressive-sampling algorithm for selecting training instances in such circumstances. The proposed algorithm formes a class distribution that performs fairly well for near optimal learning.

7. Use of Soft Computing Approaches to Decision Tree Algorithm

Decision tree algorithm requires enhancement pertaining to different problems. The problems and use of soft computing techniques as a solution are discussed here.

7.1 Use of Evolutionary Techniques

A decision trees is called optimal if it correctly classifies the data set and has minimal number of nodes. The decision tree algorithms use local greedy search method by means of information gain as target function to split the data set. The decision trees generated by these methods are efficient with classification accuracy but they often experience the disadvantage of excessive complexity. Construction of optimal decision tree is identified as NP-Complete problem [3]. This fact leads the use of genetic algorithms that provide global search through space in many directions simultaneously. The genetic algorithm is used to handle combinatorial optimization problems. Different authors have proposed a use of methodologies that integrates genetic algorithms and decision tree learning in order to evolve optimal decision trees. Although the methods are different the goal is to obtain optimal decision trees.

A. Papagelis and D. Kalles [96] proposed GATree, a genetically evolved decision trees. The genetic algorithms use binary string as initial populations but GATree uses binary decision trees as initial populations. A binary decision tree that includes one decision node with two different leaves. Initially to construct such initial trees a random attribute is selected, if that attribute is nominal valued one of its possible values is randomly selected and in case of continuous attributes an integer value from its minimum to maximum range is randomly selected. Thus the size of the search space is reduced. Two arbitrary nodes from population of subtrees are selected and nodes of those subtrees are swapped to perform crossover operation. In view of the fact that a predicted class value depends just on leaves, the crossover operator does not affect the decision trees consistency. An arbitrary node of a preferred tree is selected and it substitutes the node's test-value with a new arbitrary chosen value to perform mutation. In case if the arbitrary node is a leaf, it substitutes the installed class with a new arbitrary chosen class. Validation is performed after crossover and mutation to get final decision tree. The fitness function for evaluation is percentage of correctly classified instances on the test data set by the decision tree. The results show compact and equally accurate decision trees as compared to standard decision tree algorithms.

Similarly Z. Fu proposed GAIT [97] algorithm. The algorithm constructs a set of different decision trees from different subsets of the original data set by using a decision tree algorithm C4.5, on small samples of the data. The genetic algorithm uses these trees as its initial populations. The selection operation selects decision trees from pool by random selection mechanism. The crossover operation exchanges subtrees between the parent trees whereas mutation exchanges subtrees or leaf inside the same tree. The fitness criterion for evaluation is the classification accuracy. The validation on fitness function is performed after crossover and mutation to get final decision tree that are smaller in size. Similar approaches are proposed and experimented in [98], [99].

7.2 Use of Neural Networks

Neural networks can be used to enhance the decision tree learning. Zhou and Jiang [106] proposed a variation of C4.5 decision tree algorithm named NeC4.5 that utilizes neural network ensemble to preprocess the training data for decision tree construction. The training set may contain noise and thus the classification accuracy of the data set is reduced.

The algorithm trains a neural network ensemble and the trained ensemble is used to produce a new training set. It substitutes the preferred class labels of the original training tuples with the output from the trained ensemble. Some extra training tuples produced by the trained ensemble are also added to the new training set. The new training set is used for training C4.5. The processed training data by neural network improves classification accuracy of the decision tree classifier.

7.3 Fuzzy Decision Tree

The fuzzy decision tree provides elevated comprehensibility and the elegant performance of fuzzy systems. Fuzzy sets and fuzzy logic permits the modelling of language related uncertainties. In addition to this, it provides a symbolic outline for knowledge comprehensibility, the capability to represent fine knowledge details and the ability in dealing with problems of noise and inexact data.

The tree construction procedure for fuzzy decision tree is similar to decision tree construction algorithm. The splitting criteria are based on fuzzy boundaries but the procedures for inference are dissimilar in fuzzy decision tree. The fuzzy decision trees [101] have fuzzy decisions at each branching point. It makes calculating the best split difficult to some extent if the attributes are continuous valued or multivalued. Constructing smaller fuzzy decision trees is valuable as they contain more information in internal nodes.

The enhancements to the fuzzy decision tree algorithms are as follows. Zeidler and Schlosser [102] proposed use of membership function to discretize the attributes for handling continuous valued attributes. Janikow [103] optimized the fuzzy component of a fuzzy decision tree using a genetic algorithm. Myung Won Kim et al. [104] proposed an algorithm that determines an appropriate set of membership functions for each attribute. The algorithm uses histogram analysis with application of the genetic algorithm to tune the initial set of membership functions. A fuzzy decision tree with given set of membership functions is constructed. Fajfer and Janikow [105] described bottom-up fuzzy partitioning in fuzzy decision trees, a complement of top down technique. The proposed algorithm is useful to partition continuous valued attributes into fuzzy sets. Guetova et al. [106] proposed Incremental fuzzy decision trees. The algorithm gets equivalent results to non-incremental methods.

8. Handling Large Data Set

Handling large size data on currently available computing systems is a challenging task. Approaches like parallel, scalable and Meta decision tree are disused here in brief.

8.1 Parallel and Distributed Decision Tree Algorithms

Parallelization is a renowned, conventional means to speed up classification tasks with large amounts of data and complex programs. In the data mining applications, the size of dataset is growing that leads us to find out computationally efficient, parallel and scalable algorithms with the objective to get optimal accuracy in a reasonable amount of time with parallel processors. The algorithms work in parallel using multiple processors to construct a single reliable model.

Kazuto et al. [107] explained two methods for parallelizing decision tree algorithm, intra-node and the inter-node parallelization. Intra-node parallelization practices the parallel processing in single node and Inter-node parallelization practices the parallel processing among multiple nodes. Intra-node parallelism is further classified in record parallelism, attribute parallelism and their combination. Authors have implemented and experimented these four types of parallelizing methods with four kinds of test data. The performance analysis from these experiments states that there is a relation between the characteristics of data and the parallelizing methods. The combination of various parallelizing approaches is the most effectual parallel method

Kufrin [108] proposed a framework for decision tree construction on shared and distributed memory multiprocessor. The method builds parallel decision trees that overcome limitation of serial decision tree on large-scale training data. Narlikar [109] proposed parallel structure of a decision tree-based classifier for memory-resident datasets on SMP. The structure uses two types of divide-and-conquer parallelism, intra-node parallelization and the inter-node parallelization with lightweight Pthreads. Experimental verification on large datasets signifies that the space and time performance of the tree construction algorithm scales with the data size and number of processors.

Joshi et al. [110] proposed ScalparC, a scalable parallel classification algorithm for mining large datasets with decision trees using MPI on Cray T3D system. This implementation confirms scalability and efficiency of ScalparC for wide range of training set and wide range of processors. Hall et al. [111] presented combining decision trees learned in parallel. The proposed algorithm builds decision trees with n disjoint data subsets of a complete dataset in parallel and after that converts them into rules to combine into a single rule set. The experiments on two datasets illustrate that there is enhancement of around 40% in quantity of rules generated by decision tree. Zaki et al. [112] proposed parallel algorithm for building decision tree on shared memory multiprocessors and it was verified that it achieves good speedup. Srivastava et al. [113] presented two parallel formulations for decision tree induction as synchronous tree induction and partitioned tree induction. Authors proposed a hybrid method that implements the high-quality features of these formulations. The experimental results illustrate the high speedups and scalability in processing.

Kazuto et al. [114] proposed a parallel decision tree algorithm on a PC cluster. Plain parallelization of decision tree is not efficient due to load imbalance. The proposed algorithm is a better parallel algorithm with data redistribution. The parallel algorithm's performance on benchmark data demonstrate that it provides an improvement in speed of 3.5 times, in the best case and equal performance even in the worst case.

Jin and Agrawal [115] proposed parallel decision tree construction with memory and communication efficiency. The approach achieves very low communication volume; no need to sort the training records, during the execution and combining shared memory and distributed memory parallelization. Jin et al. [116] proposed use of SMP machines with a chain of techniques that includes full replication, full locking, fixed locking, optimized full locking and cache-sensitive locking for parallelization of data mining algorithms. The results state that among full replication, optimized full locking and cache sensitive locking, there is no clear conqueror. Any of these three techniques can outperform other technique depending upon machine and dataset. These techniques perform considerably better than the other two techniques. In decision tree construction, combining different techniques is found to be critical for obtaining high performance. Li Wenlong et al. [117] proposed parallel decision tree algorithm based on combination.

Similarly distributed decision trees learning algorithms are proposed. Jie Ouyang et al. [118] proposed Chi-Square test based decision trees induction in distributed environment. Kanishka Bhaduri et al. [119] proposed distributed decision-tree induction in peer-to-peer systems. Bin Liu et al. [120] proposed data mining in distributed data environment.

8.2 Scalable Decision Trees

Advances in technologies create a large volume of data. The large amount of knowledge in this data can be utilized to improve decision-making process of an organization. Scalability is one of the important issues in decision tree learning, a brief review of scalability handled by researchers is presented here.

SLIQ [121] a fast scalable decision tree classifier adopts the presorting scheme in the growing phase of the tree that eliminates the sorting of data at each node of decision tree. The training data are sorted just once for each numeric attribute at beginning of tree growth phase. In this method a separate list for each attribute of the training data and a separate list for the class labels of each instance is created. An entry in an attribute list has two columns, first contains an attribute value and the second contains an index into the class list. An entry of the class list also has two columns, first one contains a class label and the second contains a reference to a leaf node of the decision tree. As every leaf node represents a partition of the training data on the path from the node to the root, the class list identifies the partition to which an instance belongs. The presorting process combined with a breadth first tree growing strategy enables SLIQ to scale for disk resident large data sets and can handle both numerical and categorical data. SLIQ uses gini index as splitting criteria, uses a new tree-pruning algorithm that is inexpensive and results in compact and accurate trees.

Shafer et al. [122] proposed SPRINT that provides scalability, parallelism and removes memory restriction. It achieves parallelism by its design, which allows multiple processors to work together. In this algorithm list are created as are created in SLIQUE. Initially an attribute list for each attribute in the data set is created. The entries in these lists called attribute records that consists an attribute value, a class label and the index of the record. Initial lists for continuous attributes are sorted by attribute value. If the complete data does not fit in memory, attribute lists are kept on disk. Thus memory restrictions are solved. The initial lists formed from the training set are linked with the root of the classification tree. As the algorithm executes the tree is grown and nodes are split to create new children, the attribute lists for each node are partitioned and associated with the children. The order of the records in the list is maintained while partition and thus partitioned lists never require resorting. The algorithm uses gini index as splitting criteria. The results demonstrate good scale up and speedup on large data set. The size up performance is also good because the communication costs for exchanging split points and count matrices does not change as the training set size is increased.

Gehrke et al. [123] proposed a framework called Rainforest that provides approach for implementing scalability in decision tree algorithms with large data sets. Rainforest makes refinement to some initial steps of decision tree construction algorithm. Algorithm creates only one attribute list for all categorical attributes jointly. It creates the histograms for splitting information and thus avoids additional scan. The refinement is made up to this step, afterwards remaining part conventional decision tree algorithm proceeds.

The improvements claimed are as follows. The best splitting criteria available can be exploited for classification in a scalable manner. The algorithm claims performance improvements of greater than a factor of five over the Sprint algorithm, which is the known fastest scalable classification algorithm. Alsabti et al. proposed CLOUDS [124] a decision tree classifier for large datasets. The proposed algorithm samples for splitting points on numeric attributes followed by estimate procedure to narrow search space of best split. CLOUDS reduces computational and I/O complexity as compared to benchmark classifiers with quality in terms of accuracy and tree size. Gehrke et al. proposed [125] BOAT an approach for optimistic decision tree construction. It uses small subset of data for initial decision tree construction and refines it to construct final tree. With only two scans of training data it can construct several levels of decision tree and thus it is claimed to be faster by factor three. It can handle insertion and deletion of data in dynamic databases and thus it is first scalable incremental decision tree approach.

8.3 Meta Decision Trees

Meta learning [132] technique integrates distinct learning processes. Several meta-learning methods are been proposed for integrating autonomously learned classifiers in a parallel or distributed computing environment.

The process of constructing meta classifiers can be divided into two sub-processes. First process is to build a diverse set of base-level classifiers and second process is to combine predictions by base-level classifiers. Several approaches can be used to generate base-level classifiers to single data set. The approaches are using different learning algorithms or using a single learning algorithms. Voting, stacking and cascading are combining techniques. Meta decision tree combines multiple base level decision tree classifiers. The Meta decision tree leaf points the base level classifier, whereas ordinary decision tree leaves specify classification.

Todorovski and Dzeroski [127], [128] modified C4.5 to develop algorithm MLC4.5 for learning Meta decision tree.

Todorovski and Dzeroski compared meta decision trees with ensemble learning methods bagging and boosting and found to perform better [129]. The important issues, approaches and applications of meta-learning are discussed in [136], [137].

9. Surveys

The work on literature review of decision trees and related issues is reviewed here. Murthy [3] in his paper covered multi-disciplinary existing work on decision trees. The basics and terminologies of decision trees construction, details of tree construction methods are reviewed. The paper provides summarized existing surveys and also discusses work on determining splits at tree nodes, various pruning techniques. The other sections on several different topics relevant to tree construction such as sample size and dimensionality, work on improving greedy induction, incorporating costs, estimating probabilities from decision trees are provided. The work also compares tree based data exploration to alternative methods such as multivariate statistical methods and neural networks. At the end some recent, real-world applications of decision trees are discussed.

Safavian and Landgrebe [132] presented a survey of available design approaches for decision tree classifier. The paper provides potential advantages of decision tree classifier over single-state classifiers. The observations concerning the relation between decision trees and neural networks are also presented. This review also presents the different issues in decision tree classifiers along with the probable lacunas of each method.

Rokach and Maimon [133] analyzed various issues in decision tree construction. The survey of basics of decision trees, decision tree complexity metrics, algorithmic framework for decision trees, details of various splitting criteria, pruning methods are presented. In addition the issues like missing attribute values, misclassification cost, various decision tree inducers, handling large database with decision trees are presented.

10. Other Issues

Induction of incremental decision trees and oblique decision trees are some of the important issues in decision tree learning and these issues are discussed here.

10.1 Incremental Tree Induction

The normal decision tree-learning algorithm builds decision tree on currently available training set. If new training cases are offered, the previously built decision trees are not useful and we have to execute the decision tree algorithm once again from scratch to add new cases in decision tree. Utgoff [134] developed incremental decision tree learning algorithm. The incremental decision tree algorithm adds new training instances to current decision tree without reconstruction of the decision tree. The results produced are equivalent to the standard decision tree learning algorithms.

Reynolds and Shehri [135] proposed use of cultural algorithm based evolutionary programming to direct incremental decision tree learning. The proposed algorithm constructs a tree with minimum number of nodes and uses of few variables for its construction. It is found that evolutionary approaches enhance the quality of built trees over incremental tree approach ITI in certain cases.

10.2 Oblique Decision Tree

The Oblique decision tree is proposed for applications where the instances have numeric attribute values. The oblique classifier builds decision trees that contain linear combinations of one or more attributes at each internal node. Most of the decision tree induction algorithms generate tests at each internal node that involve a single attribute of the data at each internal node the trees. These tests are equivalent to hyperplanes that are parallel to one of the axes in the attribute domain and the resulting trees are called as axis-parallel trees. The oblique decision tree partitions the space of examples with both oblique and axis-parallel hyperplanes. The oblique hyperplanes are more complicated than finding axis-parallel

Sr. No.	Data Set	No. of Instances	No. of Attributes	Description
1	Annealing	798	38	Annealing process data
2	Australian	690	14	Credit card data
3	Balance scale	625	5	Weight and Distance data
4	Breast	286	9	Breast cancer data
5	Breast W	699	10	Breast cancer data
6	Pima Indians	768	9	Pima Indians Diabetes test data
7	German	1000	20	Credit data
8	Glass	214	10	Glass identification data
9	Heart	270	13	Heart dieses data
10	Hepatitis	155	19	Hepatitis data
11	Horse Colic	368	27	Horse Colic data
12	Iris	150	4	Iris plants data
13	Kr-vs-kp	3196	36	Chess data
14	Labor	57	16	Labor negotiation, agreements
15	Mushroom	8124	22	Physical characteristic of mushroom
16	Segment	2310	19	Hand segmented image data
17	Tic-tac-toe	958	9	Tic-tac-toe games data

Table1: Commonly used benchmark datasets in decision tree research

partitions, demanding greater computational effort.

Murthy et al. [136] extended the work of Breiman et al. [9] and developed OC1 a randomized algorithm for inducing oblique decision tree. The use of randomized hill-climbing algorithm in OC1 is more efficient than other existing randomized methods. The algorithm C4.5 uses goodness measure, which should be maximum, while OC1 uses impurity that should be minimum. The available impurity measures in OC1 are information gain, gini index, twoing rule, max minority, sum minority and sum of variations. These measures can be used as per the need of the application. Twoing rule is the default impurity measure. OC1 uses the reciprocal of twoing value as a goodness measure hence tries to minimize it and uses cost complexity pruning as default pruning method. Setiono and Liu [137] proposed simple method to generate oblique decision trees. Iyengar [138] proposed a new method of constructing oblique decision trees. The method can be integrated with ease into available decision tree tools. In this method the normal decision trees are pruned. The pruned decision trees are used to learn high-quality candidate oblique vectors. Then the learned oblique vectors are fed back to the decision tree algorithm. The resulting trees are accurate and compact. Cantu-Paz and Kamath [139] proposed evolutionary methods for inducing Oblique decision trees. The empirical results show that the EAs quickly obtain competitive classifiers and improved scale up than traditional methods to size of dataset used in training.

11. Decision Tree Learning Softwares And Commonly Used Dataset

Various decision tree softwares are available for researchers working in data mining. Some of the prominent softwares employed for analysis of data and some of the commonly used data sets for decision tree learning are discussed below.

WEKA–WEKA (Waikato Environment for Knowledge Analysis) workbench is set of different data mining tools developed by machine learning group at [140] University of Waikato, New Zealand. WEKA versions supporting windows, Linux and MAC operating systems are available. It provides various associations, classification and clustering algorithms, in addition to that it provides pre-processors like filters and attribute selection algorithms. In case of decision tree learning WEKA provides J48 i.e. C4.5 implementation in java, Simple CART and Random Forest Tree are some of the prominent tree classifiers. In J48 we can construct trees with EBP, REP and unpruned trees. The input data file is in **.**arff (attribute relation file format) format. The source code is available to the user.

C4.5- Version C4.5.8 developed by Quinlan [2] supports Unix based operating systems only**.** The package consists of programs for decision tree generator, the rule generator, the decision tree interpreter and the production rule interpreter. The decision tree generator expects two input files with extentions **.**name and **.**data file as input files. The **.**name file provides information about attributes and classes. The **.**data file contains actual attribute values with their class. The source code of C4.5 is available to the user.

OC1- OC1 is an oblique decision tree classifier by Murthy [136]. Various splitting criteria are also available with this package. The OC1 software can also be used to create both standard axis-parallel decision trees and oblique trees. The source code is available to the user.

GATree- GATree is evolutionary decision tree by Papagelis and Kalles [96]. GATree works on windows operating system. The evaluation version of GATree is available on request to the authors. Here we can set various parameters like generations, populations, crossover and mutation probability etc. to generate decision trees.

Commonly Used Data Sets

Researchers have used various benchmark datasets from University of California machine learning repository, UCI Repository [141]. Some of the most commonly used datasets by researchers are explained in Table 1 with number of records and attributes in each data set and description of data. Some of the data sets may not be available at present.

12. The Applications of Decision Trees in Various Areas

The decision tree algorithm has applications in all walks of life. The application areas are listed below

Business: Virine and Rapley [142] proposed use of decision trees in visualization of probabilistic business models. Yang et al. [143] proposed use of decision tree in customer relationship management. Zhang et al. [144] proposed use of decision tree in credit scoring for credit card users.

E-Commerce: A good online catalog is essential for the success of an e-commerce web site, Sung et al. [145] mentioned use of decision tree for construction of online catalog topologies.

Energy Modelling: Energy modelling for buildings is one of the important tasks in building design. Zhun et al. [146] proposed decision tree method for building energy demand modelling.

Image Processing: Macarthur et al. [147] proposed use of decision tree in content-based image retrieval. Park et al. [148] proposed perceptual grouping of 3-D features in aerial image using decision tree classifier.

Intrusion Detection: Sinclair et al. [149] proposed decision trees with genetic algorithms to automatically generate rules for an intrusion detection expert system. Abbes et al. [150] proposed protocol analysis in intrusion detection using decision tree.

Medical Research: Medical research and practice are the important areas of application for decision tree techniques. Stasis et al. [151] proposed decision trees algorithms for heart sound diagnosis. Lenic et al. [152] focused on decision tree methods that can support physicians in medical diagnosing in case of mitral valve prolapse. Kokol et al. [153] introduced decision trees as part of intelligent systems that help physicians. Dong et al. [154] proposed evaluating skin condition using a decision tree. Kennedy and Adams [155] proposed decision tree to help out in selecting a brain computer interface device for patients who are cognitively intact but unable to move or communicate. Hui and GaiLiping [156] proposed analysis of complex diseases by statistical estimation of diagnosis with genetic markers based on decision tree analysis.

Intelligent Vehicles: The job of finding the lane boundaries of the road is important task in development of intelligent vehicles. Gonzalez and Ozguner [157] proposed lane detection for intelligent vehicles by using decision tree.

Object Recognition: Freixenet et al. [158] proposed use of decision trees for color feature selection in object recognition for outdoor scenes.

Reliability Engineering: Claudio and Rocco [159] proposed approximate reliability expressions for network reliability using a decision tree approach. Assaf and Dugan [160] proposed method that establishes a dynamic reliability model and generates a diagnostic model using decision trees for a system.

Remote Sensing: Remote sensing is a strong application area for pattern recognition work with decision trees. Simard et al. [161] proposed decision tree-based classification for land cover categories in remote sensing. Palaniappan et al. **[162]** proposed **binary tree with genetic algorithm for land cover classification.**

Space Application: The portfolio allocation problem is pervasive to all research activities. The use of past experience in this area with help of decision trees is recommended. Manvi et al. [163] suggested use of decision trees in NASA space missions.

Speech Recognition: Amit and Murua [164] proposed speech recognition using randomized decision trees. Bahl et al. [165] proposed a tree-based statistical language model for natural language speech recognition, which predicts the next word spoken, based on previous word spoken. Yamagishi et al. [166] proposed decision trees for speech synthesis.

Software Development: Selby and Porter [167] proposed decision trees for software resource analysis. Khoshgoftaar et al. [168] proposed **decision trees for software quality classification.**

Steganalysis: Geetha et al. [169] proposed evolving decision tree based system for audio stego anomalies.

Text Processing: Diao et al. [170] introduced decision trees for text categorization.

Traffic and Road Detection: Wu et al. [171] proposed use of decision tree in analysing, predicting and guiding the traffic flow. Jeong and Nedevschi [172] proposed intelligent road region detection based on decision tree in highway and rural way environments.

Video Processing: Jaser et al. [173] proposed automatic sports video classification with decision tree. Cen and Cosman [174] explained decision trees for error concealment in video decoding.

Web Applications: Bonchi et al. [175] proposed decision trees for intelligent web caching. Chen et al. [176] presented a decision tree learning approach to diagnosing failures in large Internet sites.

REFERENCES

1. Mitchell, (1997). *Machine Learning,* The McGraw-Hill Companies, Inc.

2. J. R. Quinlan, (1993). *C4.5: Programming for Machine Learning*. San Francisco, CA: Morgan Kaufman.

3. S. K. Murthy (1998). Automatic construction of decision trees from data: a multi-disciplinary survey. *Data Mining and Knowledge Discovery,* Vol. 2, No. 4, pp. 345-389.

4. E. Alpaydin (2005). *Introduction to machine Learning* Prentice-Hall of India.

5. S. Ruggieri (2002). Efficient C4.5. *IEEE Transaction on Knowledge and Data Engineering,* Vol.14, No. 2, pp. 438-444.

6. Moshe Ben-Bassat (1987). Use of distance measure, Information measures and error bounds on feature evaluation. In Sreerama Murthy (1), pp. 9-11.

7. Mark Last and Oded Maimon (2004).. A compact and accurate model for classification. *IEEE Transactions on Knowledge and Data Engineering,* Vol. 16, No. 2, pp. 203-215.

8. Byung Hwan Jun, Chang Soo Kim, Hong-Yeop Song and Jaihie Kim (1997). A new criterion in selection and discretization of attributes for the generation of decision trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 19, No. 12, pp. 1371-1375.

9. Leo Breiman, Jerome H. Friedman, Richard A. Olshen, and Charles J. Stone (1984). *Classification and Regression Trees.* Wadsworth International Group, Belmont, California.

10. S. K. Murthy, Simon Kasif and Steven Salzberg (1994). A system for induction of oblique decision trees. *Journal of Artificial Intelligence Research 2*, pp.1-33.

11. R. S. Mantaras (1991). A distance based attribute selection measure for decision tree induction. *Technical Report, Machine Learning,* Vol. 6, pp. 81-92.

12. B. Chandra, R. Kothari, P. Paul (2010). A new node splitting measure for decision tree construction Pattern *Recognition* Vol.43, Elsevier Publishers, pp. 2725-2731.

13. E. Rounds (1980). A combined nonparametric approach to feature selection and binary decision tree design. *Pattern Recognition,* Vol. 12, pp. 313-317.

14. P. E. Utgoff and J. A. Clouse (1996). A Kolmogorov-Smirnoff metric for decision tree induction. *Tech. Rep. No. 96-3*, Dept. Comp. Science, University Massachusetts, Amherst.

15. J. K. Martin (1997). An exact probability metric for decision tree splitting and stopping. *Machine Learning,* Vol. 28, No. 2-3, pp. 257-29.

16. W. L. Buntine and T. Niblett (1992). A further comparison of splitting rules for decision-tree induction. *Machine Learning*, Vol. 8, pp. 75-85.

17. T. Windeatt and G. Ardeshir (2001). An empirical comparison of pruning methods for ensemble classifiers. *Proc. of 4th International Conference on Advances in Intelligent Data Analysis,* Cascais, Portugal, pp. 208-217.

18. Floriana Esposito, Donato Malerba and Giovanni Semeraro (1997). A comparative analysis of methods for pruning decision trees. *IEEE Transactions On Pattern Analysis And Machine Intelligence*, Vol. 19, No. 5, pp. 476-491.

19. J. R. Quinlan (1987). Simplifying decision trees. *International Journal of Man Machine Studies* Vol. 27, pp. 221-234.

20. J. Mingers (1989). An empirical comparison of pruning methods for decision tree induction,'' *Machine Learning*, Vol.3, pp. 227-243.

21. M. Mehta, J. Rissanen and R. Agrawal (1995). MDL-based decision tree pruning. *Proc. of the 1st International Conference on Knowledge Discovery in Databases and Data Mining*, Montreal, Canada, pp. 216-221.

22. J. Ross Quinlan and Ronald L. Rivest (1989). Inferring decision trees using the minimum description length principle. *Inform. Comput.* Vol. 80, pp. 227-248.

23. I. Bratko and M. Bohanec (1994). Trading accuracy for simplicity in decision trees. *Machine Learning*, Vol. 15, pp. 223-250.

24. H. Allamullim (1996). An efficient algorithm for optimal pruning of decision trees. *Artificial Intelligence*, Vol. 83, Issues 2, pp. 347-362.

25. Matti Kaariainen (2004). Learning small trees and graphs that generalize. A Report, University of Helsinki, Finland Series of Publications Helsinki.

26. Lawrence Hall, Richard Collins, Kevin W. Bowyer and Robert Banfield (2002). Error-Based pruning of decision trees grown on very large data sets can work! *Proc. of the 14th IEEE International Conference on Tools with Artificial Intelligence*, pp. 233-238.

27. T. Oates and D. Jensen (1999). Toward a theoretical understanding of why and when decision tree pruning algorithms fail. *Proc. of the Sixteenth National Conference on Artificial Intelligence*, pp. 372-378.

28. Jan Macek and Lenka Lhotsk (2004). Gaussian complexities based decision tree pruning. *Cybernetics and Systems 2004*, Austrian Society for Cybernetics Studies Vienna, pp. 713-718.

29. Eibe Frank (2000). Pruning Decision Trees and List. A Doctoral Thesis Submitted to University of Waikato.

30. J. P. Bradford, Clayton Kunz, Ron Kohavi, Clifford Brunk and C. E. Brodley (1998). Pruning decision trees with misclassification costs. *European Conference on Machine Learning*, pp. 131-136.

31. Clayton Scott (2005). Tree pruning with subadditive penalties. *IEEE Transactions On Signal Processing*, Vol. 53, No. 12, pp. 4518-4525.

32. Andrew P. Bradley and Brian C. Lovell (1995). Cost-sensitive decision tree pruning: use of the ROC curve. *In Eighth Australian Joint Conference on Artificial Intelligence*, November 1995, Canberra, Australia pp. 1-8.

33. J. Cai1, J. Durkin1 and Q. Cai (2005). CC4.5: cost-sensitive decision tree pruning. *Proc. of Data Mining Conference*, Skiathos, Greece. pp. 239-245.

34. Y. Mansour (1997). Pessimistic decision tree pruning based on tree size. *Proc. of 14th International Conference on Machine Learning*, pp. 195-201.

35. X. Huo, Seoung Bum Kim, Kwok-Leung Tsui and Shuchun Wang (2006). FBP: A frontier-based tree-pruning algorithm. *INFORMS Journal on Comp*uting Vol. 18, No. 4, pp. 494-505.

36. Johannes Faurnkranz (1997). Pruning algorithms for rule learning. *Machine Learning*, Vol. 27, pp. 139-172.

37. Shesha Shah and P. S. Sastry (1999). New algorithms for learning and pruning oblique decision trees. *IEEE Transactions On Systems, Man, And Cybernetics—Part C: Applications And Reviews*, Vol. 29, No. 4, pp. 494-505.

38. G.V. Kass (1980). An exploratory technique for investigating large quantities of categorical data. *Applied Statistics*, Vol. 29, No.2, pp.119-127.

39. Quinlan J. R. (1986). Induction of decision trees. *Machine Learning*, Vol.1-1, pp. 81-106.

40. Ron Kohavi (1994). Feature subset selection as search with probabilistic estimates. *In proc. the AAAI Fall Symposium on Relevance.* pp.122-126.

41. Rich Caruana and Dayne Freitag (1994). Greedy attribute selection. *Proc. of the 11th International Conference on Machine Learning.* pp.28-36.

42. Mark Last, Abraham Kandel, Oded Maimon and Eugene Eberbach (2000). Anytime algorithm for feature selection. *Proc. of Second International Conference on Rough Sets and Current Trends in Computing*, pp. 532 - 539.

43. Shaomin Wu and Peter A. Flach (2002). Feature selection with labelled and unlabelled data. In Marko Bohanec, Dunja Mladenic, and Nada Lavrac, editors, *ECML/PKDD'02 workshop on Integrating Aspects of Data Mining, Decision Support and Meta-Learning*. pp.156-167.

44. Huang Yuan, Shian-Shyong Tseng, Wu Gangshan and Zhang Fuyan (1999). A two-phase feature selection method using both filter and wrapper. *Proc. of IEEE International Conference on Systems, Man and Cybernetics*, pp. 132 - 136.

45. Krzysztof Grabczewski and Norbert Jankowski (2005). Feature selection with decision tree criterion. *Proc. of Fifth International Conference on Hybrid Intelligent Systems.*6-9 Nov. pp.212-217.

46. Jose Bins, Bruce A. Draper (2001). Feature selection from huge feature sets. *Proc. of International Conference on Computer Vision*, Vancouver, pp. 159-165.

47. Cesar Guerra-Salcedo, Stephen Chen, Darrell Whitley and Stephen Smith (1999). Fast and accurate feature selection using hybrid genetic strategies. *Proc. of the Congress on Evolutionary Computation.* pp. 177-184.

48. Jacques-Andre Landry, Luis Da Costa and Thomas Bernier (2006). Discriminant feature selection by genetic programming: towards a domain independent multi-class object detection system. *Journal of Systemics, Cybernetics and Informatics*, Vol. 1, 3. pp. 76-81.

49. Bala, J. Huang and H. Vafaie, K. DeJong and H. Wechsler (1995). Hybrid learning using genetic

algorithms and decision trees for pattern classification. *Proc. of the IJCAI conference, Montrea*l.pp.719-724

50. Gaelle Legrand and Nicolas Nicoloyannis (2005). Feature selection and preferences aggregation. *Machine Learning and Data Mining in Pattern Recognition,* Springer Heidelberg, pp. 203-217.

51. Mark A. Hall and Lloyd A. Smith (1997). Feature subset selection: a correlation based filter approach. *Proc. of International Conference on Neural Information Processing and Intelligent Information Systems1997,* pp. 855-858.

52. W. Duch, J. Biesiada, T. Winiarski, K. Grudzinski and K. Gr. Abczewski (2002). Feature selection based on information theory filters and feature elimination wrapper methods. *Proc. of the International Conference on Neural Networks and Soft Computing, Advances in Soft Computing,* pp. 173-176.

53. Mark A. Hall (2000). Correlation-based feature selection for discrete and numeric class machine learning. *Proc. of International Conference on Machine Learning, Stanford University,* CA. Morgan Kaufmann Publishers, pp. 359-366.

54. Huang Yuan, Shian-Shyong Tseng, Wu Gangshan and Zhang Fuyan (1999). A two-phase feature selection method using both filter and wrapper. *Proc. of IEEE International Conference on Systems, Man and Cybernetics,* pp. 132 - 136.

55. Pier Luca Lanzi (1997). Fast feature selection with genetic algorithms: a filter approach. *Proc. of 1997 IEEE International Conference on Evolutionary Computation.* pp.537-540

56. G. H. John (1995). Robust decision trees: Removing outliers from databases. In *Proc. of the First ICKDDM,* 1995, pp. 174-179.

57. A. Arning, R. Agrawal, and P. Raghavan. A linear method for deviation detection in large databases. In *KDDM* 1996, Pages 164–169.

58. A. I. Guyon, N. Matic and V. Vapnik, "Discovering informative patterns and data cleaning", Advances in knowledge discovery and data mining, AAAI, 1996, Pages: 181 – 203.

59. G D.Gamberger and N. Lavrac. Conditions for Occam's Razor applicability and noise elimination. In Marteen van Someren and Gerhard Widmer, editors, Proceedings of the 9th European Conference on Machine Learning, Springer, 1997. Pages 108-123.

60. E. M. Knorr and R. T. Ng. "A unified notion of outliers: properties and computation", In Proceedings of 3rd International Conference on Knowledge Discovery and Data Mining, 1997.

61. E. M. Knorr and R. T. Ng. "Algorithms for mining distance-based outliers in large datasets." In *Proc. 24th VLDB,* 1998, Pages 392–403, 24–27.

62. D. Tax and R. Duin, "Outlier detection using classifier instability" Proceedings of the workshop Statistical Pattern Recognition, Sydney 1998.

63. C. E. Brodley and M. A. Friedl (1999). Identifying mislabeled training data. *Journal of Artificial Intelligence Research 11,* pp. 131-167.

64. I. S. Weisberg (1985) Applied Linear Regression, John Wiley and Sons.

65. D. D. Gamberger, N. Lavrac, and C. Groselj, "Experiments with noise filtering in a medical domain.", In *Proc. 16th ICML*, Morgan Kaufman, San Francisco, CA, 1999, Pages 143–151.

66. S. Schwarm and S. Wolfman, "Cleaning data with Bayesian methods" 2000. Final project report for University of Washington Computer Science and Engineering CSE574, March 16, 2000.

67. S. Ramaswamy, R. Rastogi, and K. Shim. "Efficient algorithms for mining outliers from large data sets." ACM SIGMOD Volume 29, Issue 2 June 2000, Pages: 427 - 438.

68. V. Raman and J.M. Hellerstein, "An interactive framework for data transformation and cleaning" Technical report University of California Berkeley, California, September 2000.

69. J. Kubica and A. Moore, "Probabilistic noise identification and data cleaning", Third IEEE International Conference on Data Mining, 19-22 Nov. 2003.

70. V. Verbaeten and A. V. Assche "Ensemble methods for noise elimination in classification problems. In *Multiple Classifier Systems*." Springer, 2003.

71. J. A. Loureiro, L. Torgo, and C. Soares, "Outlier detection using clustering methods: a data cleaning application", in Proceedings of KDNet Symposium on Knowledge-based Systems for the Public Sector. Bonn, Germany, 2004.

72. H. Xiong, G. Pande, M. Stein and Vipin Kumar, "Enhancing Data analysis with noise removal", IEEE Transaction on knowledge and Data Engineering Volume 18, Issue 3, March 2006, Pages: 304 – 319.

73. Seung Kim , Nam Wook Cho , Bokyoung Kang , Suk-Ho Kang, "Fast outlier detection for very large log data" *Expert Systems with Applications* Vol. 38, 2011 Elsevier. pp. 9587–9596.

74. S. Hido, Y. Tsuboi, H. Kashima, M. Sugiyama and T. Kanamori Statistical Outlier Detection Using Direct Density Ratio Estimation. Knowledge and Information Systems. vol. 26, no.2, pp.309-336, 2011.

75. George John and Pat Langley, "Static Versus Dynamic Sampling for Data Mining", *In Proceedings of the Second International Conference on Knowledge Discovery and Data Mining* 1996, AAAI Press, pp. 367-370.

76. Foster Provost and David Jensen and Tim Oates, "Efficient Progressive sampling", *In Proceedings of the Fifth International Conference on Knowledge Discovery and Data Mining,* 1999 ACM Press, pp.23-32.

77. D. V. Patil and R.S. Bichkar (2006). A hybrid evolutionary approach to construct optimal decision trees with large data sets. *In Proc. IEEE ICIT06 Mumbai,* 15-17 December 2006, pp. 429-433.

78. R .J. Little and D.B. Rubin (1987). *Statistical Analysis with Missing Data.* John Wiley and Sons, New York.

79. G. Batista and M.C. Monard (2003). An analysis of four missing data treatment methods for supervised learning. *Applied Artificial Intelligence*, Vol. 17, pp. 519-533.

80. Jerome H. Friedman, Jon Louis Bentley and Raphael Ari Finkel (1977). An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematical Software,* Vol. 3 pp. 209-226.

81. J. R. Quinlan (1986). Unknown attribute values in induction. *Journal of Machine Learning* Vol. 1, pp. 81-106.

82. Kuligowski R. J. and Barros A. P. Using artificial neural Networks to estimate missing rainfall data. Journal AWRA 34(6), 14.1998.

83. Brockmeier L. L., Kromrey J. D. and Hines C. V., 1998.Systematically Missing Data and Multiple Regression Analysis: An Empirical Comparison of Deletion and Imputation Techniques. *Multiple Linear Regression Viewpoints*, Vol. 25, 20-39.

84. Abebe A. J., Solomatine D. P. and Venneker R. G. W. 2000. Application of adaptive fuzzy rule-based models for reconstruction of missing precipitation events. Hydrological Sciences Journal.45 (3), 425–436.

85. Sinharay S., Stern H.S. and Russell D. 2001. The use of multiple imputations for the analysis of missing data. Psychological Methods Vol.4: 317–329.

86. Khalil K., Panu M. and Lennox W. C. 2001. Groups and neural networks based stream flow data infilling procedures. Journal of Hydrology, 241, 153–176.

87. Bhattacharya B., Shrestha D. L. and Solomatine D. P. 2003. Neural networks in reconstructing missing wave data in Sedimentation modeling. In the Proceedings of 30th IAHR Congress, Thessaloniki, Greece Congress, August 24-29 2003 Thessaloniki, Greece.

88. Fessant F. and Midenet, S. 2002. Self-organizing map for data imputation and correction in surveys. Neural Comput. Appl. 10, 300–310.

89. Musil C. M., Warner C. B., Yobas P. K. and Jones S. L. 2002. A comparison of imputation techniques for handling missing data. Weston Journal of Nursing Research 24(7), 815–829.

90. Junninen H., Niska H., Tuppurainen K., Ruuskanen J. and Kolehmainen M. 2004. Methods for imputation of missing values in air quality data sets. Atoms. Environ. 38, 2895–2907.

91. M. Subasi, E. Subasi and P.L. hammer, 2009. New Imputation Method for Incomplete Binary Data, Rutcor Research Report, August 2009.

92. Amman Mohammad Kalteh and Peder Hjorth, 2009. Imputation of Missing values in precipitation-runoff process database. Journal of Hydrology research.40.4, pages 420- 432.

93. Rhian M. Daniel, Michael G. Kenward, “A method for increasing the robustness of multiple imputation,” *Computational Statistics and Data Analysis*, doi 10.1016/j.csda.2011.10.006, Elsevier 2011.

94. Patil and Bichkar (2010). Multiple Imputation of Missing Data with Genetic Algorithms based Techniques. International Journal on Computer Applications Special Issue on Evolutionary Computation in Optimisation Techniques (2), pp. 74 - 78.

95. Gary Mitchell Weiss (2003). The Effect of Small Disjuncts and Class Distribution on Decision Tree Learning. A Doctoral Thesis Submitted to the Graduate School, New Brunswick Rutgers, The State University of New Jersey.

96. A. Papagelis and D. Kalles (2000). GATree: Genetically evolved decision trees. *Proc. 12th International Conference On Tools With Artificial Intelligence*, pp. 203-206.

97. Zhiwei Fu and Fannie Mae (2001). A computational study of using genetic algorithms to develop intelligent decision trees. *Proc. of the 2001 IEEE Congress On Evolutionary Computation* Vol. 2. pp. 1382-1387.

98. A. Niimi and E. Tazaki (2000). Genetic programming combined with association rule algorithm for decision tree construction. *Proc. of fourth International Conference on Knowledge-Based Intelligent Engineering Systems and Allied Technologies,* Vol. 2, pp. 746-749.

99 Y. Kornienko and A**.** Borisov (2003). Investigation of a hybrid algorithm for decision tree generation. *Proc. of the Second IEEE International Workshop on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications,* pp. 63-68.

100 Zhi-Hua Zhou and Yuan Jiang (2004). NeC4.5: Neural ensemble based C4.5. *IEEE Transactions On Knowledge And Data Engineering,* Vol. 16, No. 6. pp. 770-773.

101 C. Z. Janikow (1998). Fuzzy decision trees: Issues and methods. *IEEE Transactions on Systems, Man, and Cybernetics,* Vol. 28, Issue 1, pp. 1-14.

102 Zeidler and M. Schlosser (1996). Continuous-valued attributes in fuzzy decision trees. *Proc. of the International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems,* pp. 395-400,

103 C. Z. Janikow (1996). A genetic algorithm method for optimizing the fuzzy component of a fuzzy decision tree. *In CA for Pattern Recognition*, editors S. Pal and P. Wang, CKC Press, pp. 253-282,

104 Myung Won Kim, Joong Geun Lee, and Changwoo Min (1999). Efficient fuzzy rule generation based on fuzzy decision tree for data mining. *Proc. of IEEE International Fuzzy Systems Conference* Seoul, Korea, 22-25.

105 Maciej Fajfer and C. Z. Janikow (2000). Bottom-up fuzzy partitioning in fuzzy decision trees. *Proc. of 19th International Conference of the North American Fuzzy Information Processing Society, 2000* pp. 326 - 330.

106 Marina Guetova, Steffen Holldobler and Hans-Peter

Storr (2004). Incremental fuzzy decision trees. *International Conference on Fuzzy Sets and Soft Computing in Economics and Finance*, St. Petersburg, Russia.

107 Kazuto Kubota, Hiroshi Sakai, Akihiko Nakase and Shigeru Oyanagi (2000). Parallelization of decision tree algorithm and its performance evaluation. *Proc. of The Fourth International Conference on High Performance Computing in the Asia-Pacific Region,* Vol. 2. pp. 574 -579.

108 R. Kufrin (1997). Decision trees on parallel processors Machine *Intelligence and pattern recognition ,* Vol. 20, Elsevier . pp. 279-306.

109 G. J. Narlikar (1998). A parallel, multithreaded decision tree builder. *A Technical Report,* School of Computer Science, Carnegie Mellon University.

110 M. V. Joshi, G. Karypis and V. Kumar (1998). Scalparc: A new scalable and efficient parallel classification algorithm for mining large datasets. *Proc. of the International Parallel Processing Symposium.* pp. 573-579.

111 Lawrence O. Hall, Nitesh Chawla and Kevin W. Bowyer (1998). Combining decision trees learned in parallel. *Distributed Data Mining Workshop at International Conference of Knowledge Discovery and Data Mining*. pp. 77-83.

112 M. J. Zaki, C.T. Ho and R. Agrawal (1999). Parallel classification for data mining on shared-memory multiprocessors. *IEEE International Conference on Data Engineering,* pp. 198-205.

113 Anurag Srivastava, EuiHong Han, Vipin Kumar and Vineet Singh (1999). Parallel formulations of decision-tree classification algorithms. *Data Mining and Knowledge Discovery: An International Journal*, vol. 3, no. 3. pp. 237-261.

114. Kazuto Kubota, Akihiko Nakase and Shigeru Oyanagi (2001). Implementation and performance evaluation of dynamic scheduling for parallel decision tree generation. *Proc. of the 15th International Parallel and Distributed Processing Symposium*. pp. 1579-1588.

115. Ruoming Jin and Gagan Agrawal (2003). Communication and memory efficient parallel decision tree construction. *Proc. of Third SIAM Conference on Data Mining*.

116. Ruoming Jin, Ge Yang and Gagan Agrawal (2004). Shared memory parallelization of data mining algorithms: techniques, programming interface, and performance. *IEEE Transactions On Knowledge And Data Engineering,* Vol. 16, No. 10. pp.71-89.

117. Li Wenlong, Xing Changzheng, "Parallel Decision Tree Algorithm Based on Combination", IEEE *International Forum on Information Technology and Applications (IFITA) Kunming,* 2010, 16-18 July 2010, pp. 99-101.

118. Jie Ouyang, Patel N., Sethi I.K., Chi-Square Test Based Decision Trees Induction in Distributed Environment *IEEE International Conference on Data Mining Workshops,* 2008. ICDMW '08. 15-19 Dec. 2008 pp. 477 – 485.

119. Kanishka Bhaduri, Ran Wolff, Chris Giannella, Hillol Kargupta, "Distributed Decision-Tree Induction in Peer-to-Peer Systems", *Journal Statistical Analysis and Data Mining.* Vol. 1 Issue 2, June 2008 John Wiley and Sons 2008.

120. Bin Liu, Shu-Gui Cao, Xiao-Li Jim, Zhao-Hua Zhi, "Data mining in distributed data environment", *International Conference on Machine Learning and Cybernetics (ICMLC),* 11-14 July 2010 Vol.1 pp.421 – 426.

121. M. Mehta, R. Agrawal and J. Rissanen (1996). SLIQ: A fast scalable classifier for data mining. *Proc. of the Fifth international Conference on Extending Database Technology,* Avignon, France. pp. 18-32.

122. Shafer, R. Agrawal and M. Mehta (1996). SPRINT: A scalable parallel classifier for data mining. *Proc. of the 22nd VLDB Conference.* pp. 544-555.

123. J. Gehrke, R. Ramakrishnan and V. Ganti (1998). Rainforest—A framework for fast decision tree construction of large datasets. *Proc. of Conference on Very Large Databases (VLDB).* pp .416-427.

124. K. Alsabti, S. Ranka and V. Singh (1998). CLOUDS: a decision tree classifier for large datasets. *Proc. of Conference on Kno wledge Discovery and Data Mining (KDD-98),* pp. 2-8.

125. J. Gehrke, V. Ganti, R. Ramakrishnan and W. Loh (1999). BOAT-optimistic decision tree construction. *Proc. of Conference SIGMOD,* pp.169-180.

126. P. Chan and S. J. Stolfo (1993). Toward parallel and distributed learning by meta-learning. *In Working Notes AAAI Work. Knowledge Discovery in Databases*, pp. 227-240.

127. Todorovski L. and Dzeroski (2000). Combining multiple models with meta decision trees. *Proc. of the Fourth European Conference on Principles of Data Mining and Knowledge Discovery,* pp. 54-64.

128. L. Todorovski and Dzeroski (2003). Combining classifiers with meta decision trees. *Machine Learning*, Vol. 50, issue 3, pp. 223-249.

129. B. Zenko, L. Todorovski, and Dzeroski (2001). A comparison of stacking with meta decision trees to bagging, boosting, and stacking with other methods. *Proc. of the 2001 IEEE International Conference on Data Mining,* pp. 669-670.

130. Andreas L. Prodromidis, Philip K. Chan and Salvatore J. Stolfo (2000). Meta-learning in distributed data mining systems: Issues and approaches.editors Hillol Kargupta and Philip Chan, *Book on Advances of Distributed Data Mining* AAAI press. pp. 81-113.

131. S. Stolfo, W. Fan, W. Lee, A. Prodromidis and P. Chan (1997). Credit Card Fraud Detection Using Metalearning: Issues and Initial Results. *In working notes of AAAI Workshop on AI Approaches to Fraud Detection and Risk Management*.

132. S. Rasoul Safavian and David Landgrebe (1991). A survey of decision tree classifier methodology. *IEEE Transaction on Systems, Man, and Cybernetics.* Vol.

21, Issue 3, pp. 660 - 674.

133. Lior Rokach and Oded Maimon (2005). Top-down induction of decision trees classifiers-a survey. *IEEE Transactions On Systems, Man, And Cybernetics-Part C: Applications And Reviews,* Vol. 35, No. 4. pp. 476-487.

134. Utgoff P.E. (1989). Incremental induction of decision trees. *Machine Learning,* 4. pp. 161-186.

135. R. Reynolds and Hasan Al-Shehri (1998). The use of cultural algorithms with evolutionary programming to guide decision tree induction in large databases Proc. *of The 1998 IEEE International conference on Evolutionary Computation, at IEEE World Congress on Computational Intelligence at* Anchorage, AK, USA, pp. 441-546.

136. S. K. Murthy, S. Kasif, S. Salzberg, And R. Beigel (1993). OC1: Randomized induction of oblique decision trees. In Proc. Eleventh National Conference on Artificial Intelligence, *Washington,* DC, 11-15th, July 1993. AAAI Press, pp. 322-327.

137. Rudy Setiono and Huan Liu (1999). A connectionist approach to generating oblique decision trees. *IEEE Transactions On Systems, Man, And Cybernetics,* Vol. 29, No. 3.

138. Iyengar V. S. (1999). HOT: Heuristics for oblique trees. *Proc. of Eleventh International Conference on Tools with Artificial Intelligence,* IEEE Press, pp. 91-98.

139. Cantu-Paz E. and Kamath C. (2003). Inducing oblique decision trees with evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, Vol. 7, Issue 1, pp. 5-68.

140. Ian H. Witten and Eibe Frank (2005). *Data Mining Practical Machine Learning Tools and Techniques*. Morgan Kaufmann.

141. Frank, A. and Asuncion, A. (2010). UCI Machine Learning Repository Irvine, CA [http://archive.ics.uci.edu/ml]. University of California, School of Information and Computer Science.

142. Lev Virine and Lisa Rapley (2003). Visualization of probabilistic business models. *Proc. of the 2003 Winter Simulation Conference,* Vol. 2, pp. 1779-1786.

143. Qiang Yang, Jie Yin, Charles X. Ling and Tielin Chen (2003). Post processing decision trees to extract actionable knowledge. *Proc. of the Third IEEE International Conference on Data Mining, 2003.* Florida, USA.

144. Defu Zhang, Xiyue Zhou, Stephen C.H. Leung, Jiemin Zheng, (2010). Vertical bagging decision trees model for credit scoring. *Expert Systems with Applications,* Elsevier Publishers, Vol. 37. pp. 7838-7843.

145. Wing-Kin Sung, David Yang, Siu-Ming Yiu, David W. Cheung, Wai-Shing Ho, and Tak-Wah Lam (2002). Automatic construction of online catalog topologies. *IEEE Transactions On Systems, Man, And Cybernetics—Part C: Applications And Reviews,* Vol. 32, No. 4.

146. Zhun Yu, Fariborz Haghighat, Benjamin C.M. Fung and Hiroshi Yoshino (2010). A decision tree method for building energy demand modeling *International Journal of Energy and Buildings* Vol.42. pp. 1637-1646.

147. Sean D. MacArthur, Carla E. Brodley, Avinash C. Kak and Lynn S. Broderick (2002). Interactive content-based image retrieval using relevance feedback. *Computer Vision and Image Understanding,* pp. 55-75.

148. Kyu Park, Kyoung Mu Lee and Sang Uk Lee (1999). Perceptual grouping of 3D features in aerial image using decision tree classifier. *In Proc. of 1999 International Conference on Image Processing*, Vol. 1, pp. 31 - 35.

149. Chris Sinclair, Lyn Pierce and Sara Matzner, An application of machine learning to network intrusion detection. *In Proc. of 15th Annual Computer Security Applications Conference. pp. 371-37*

150. Tarek Abbes, Adel Bouhoula and Michael Rusinowitch (2004). Protocol analysis in intrusion detection using decision tree. *Proc. of the International Conference on Information Technology: Coding and Computing,* IEEE. pp. 404-408.

151. A. Ch. Stasis, E. N. Loukis, S. A. Pavlopoulos and D. Koutsouris (2003). Using decision tree algorithms as a basis for a heart sound diagnosis decision support system. *Proc. of the 4th Annual IEEE Conference on Information Technology Applications in Biomedicine,* UK, pp. 354 -357.

152. M. Lenic, P. Povalej, M. Zorman V. Podgorelec, P. Kokol and L. Lhotska (2003). Multimethod machine learning approach for medical diagnosing. *Proc. of the 4th Annual IEEE Conf on Information Technology Applications in Biomedicine,* UK, pp. 195-198.

153. Peter Kokol, Milan Zorman, Vili Podgorelec and Spela Hleb Babie (1999). Engineering for intelligent systems. *Proc. of 1999 IEEE International Conference on Systems, Man, and Cybernetics*, Vol. 6, pp. 306 - 311.

154. Ming Dong, Ravi Kothari, Marty Visschert and Steven B. Hoatht (2001). Evaluating skin condition using a new decision tree induction algorithm. *Proc. International Joint Conference on Neural Networks,* Vol.4, pp. 2456 - 2460.

155. P. R. Kennedy and K. D. Adams (2003). A decision tree for brain-computer interface devices. IEEE *Transactions On Neural Systems And Rehabilitation Engineering,* Vol. 11, No. 2. pp. 148-150.

156. Liu Hui and GaiLiping (2009). Statistical estimation of diagnosis with genetic markers based on decision tree analysis of complex disease International Journal of Computers in Biology and Medicine Vol. 39, pp. 989-992.

157. Juan Pablo Gonzalez and U. Ozguner (2000). Lane detection using histogram-based segmentation and decision trees. Proc. *of IEEE Intelligent Transportation Systems*, pp. 346-351.

158. J. Freixenet, X. Lladb, J. Marti and X. Cufi (2000). Use of decision trees in color feature selection. application to object recognition in outdoor scenes. *Proc. of International Conference on Image Processing,* Vol. 3, pp. 496-499.

159. Claudio M. Rocco S. (2004). Approximate reliability expressions using a decision tree approach. *Proc. of Annual Symposium - RAMS Reliability and Maintainability,* pp. 116-121.

160. Tariq Assaf and Joanne Bechta Dugan (2004). Diagnostic expert systems from dynamic fault trees. *Annual Symposium-RAMS Reliability and Maintainability,* pp. 444-450.

161. Simard, Sasan S. Saatchi and Gianfranco De Grandi (2000). The use of decision tree and multiscale texture for classification of jers-1 sar data over tropical forest. *IEEE Transactions On Geoscience And Remote Sensing,* Vol. 38, No. 5.

162. Palaniappan, Feng Zhu, Xinhua Zhuang and Yunxin Zhao Blanchard (2000). Enhanced binary tree genetic algorithm for automatic land cover classification. Proc. of *International Geoscience and Remote Sensing Symposium,* pp.688-692.

163. Ram Manavi, C. Weisbin, W. Zimmerman and G. Rodriguez (2002). Technology portfolio options for NASA missions using decision trees. *Proc. of IEEE Aerospace Conference,* Big Sky, Montana. pp. 115-126.

164. Yali Amit and Alejandro Murua (2001). Speech recognition using randomized relational decision trees. *IEEE Transactions On Speech And Audio Processing,* Vol. 9, No. 4. pp. 333-341.

165. L.R. Bahl, Peter F. Brown, Peter V. De Souza and Robert L. Mercer (1989). A tree-based statistical language model for natural language speech recognition. *IEEE Transactions On Acoustics, Speech, And Signal Processing,* Vol. 37, No. 7. pp. 1001-1008.

166. Junichi Yamagishi, Makoto Tachibana, Takashi Masuko and Takao Kobayashi, Speaking style adaptation using context clustering decision tree for hmm-based speech synthesis. Proc. of IEEE International Conference on Acoustics, Speech and Signal Processing, Vol. 1, pp. 5-8.

167. Selby R.W. and Porter (1988). A learning from examples: generation and evaluation of decision trees for software resource analysis. *IEEE Transactions On Software Engineering,* Vol. 14, pp.1743-1757.

168. T.M. Khoshgoftaar, N. Seliya and Yi Liu (2003).

Genetic programming-based decision trees for software quality classification. *Proc. of 15th IEEE International Conference on Tools with Artificial Intelligence,* pp. 374-383.

169. S.Geetha, N. N. Ishwarya, N. Kamaraj (2010). Evolving decision tree rule based system for audio stego anomalies detection based on Hausdorff distance statistics *Information Sciences* Elsevier Publisher pp. 2540-2559.

170. Lili Diao, Keyyun Hu, Yuchan Lu, Chunyi Shi Boosting (2002). Simple decision trees with Bayesian learning for text categorization. *IEEE Robotics and Automation Society Proc. of the 4th World Congress on Intelligent Control and Automation,* Shanghai, China, pp. 321- 325.

171. Bing Wu, Wen-Jun Zhou and Wei-Dong Zhang (2003). The applications of data mining technologies in dynamic traffic prediction. *IEEE Intelligent Transportation Systems,* Vol.1 pp. 396-401.

172. Pangyu Jeong and Sergiu Nedevschi (2003). Intelligent road detection based on local averaging classifier in real-time environments. *Proc. of the 12th International Conference on Image Analysis and Processing.*

173. Edward Jaser, Josef Kittler and William Christmas (2004). Hierarchical decision making scheme for sports video categorization with temporal post-processing. *Proc. of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. pp. 908-913.

174. Song Cen and Pamela C. Cosman (2003). Decision trees for error concealment in video decoding. *IEEE Transactions on Multimedia,* Vol. 5, No. 1. pp. 1-7.

175. Francesco Bonchi, Giannotti, G. Manco, C. Renso, M. Nanni, D. Pedreschi and S. Ruggieri (2001). Data mining for intelligent web caching. *Proc. of International Conference on Information Technology: Coding and computing, 2001*, pp. 599 - 603.

176. M. Chen, A. Zheng, J. Lloyd, M. Jordan and E. Brewer (2004). Failure diagnosis using decision trees. *Proc. of the International Conference on Autonomic Computing, pp. 36 -43.*