CC-SLIQ: Performance Enhancement with $2k$ Split Points in SLIQ Decision Tree Algorithm

Narasimha Prasad L V, *Member, IAENG,* and Mannava Munirathnam Naidu, *Member, IAENG*

***Abstract*—Decision trees have been found to be very effective for classification in the emerging field of data mining. This paper proposes a new method: CC-SLIQ (Cascading Clustering and Supervised Learning In Quest) to improve the performance of the SLIQ decision tree algorithm. The drawback of the SLIQ algorithm is that in order to decide which attribute is to be split at each node, a large number of Gini indices have to be computed for all attributes and for each successor pair for all records that have not been classified. SLIQ employs a presorting technique in the tree growth phase that strongly affects its ability to find the best split at a decision tree node. However, the proposed model eliminates the need to sort the data at every node of the decision tree; as an alternative the training data uses a *k*-means clustering data segmentation only once for every numeric attribute at the beginning of the tree growth phase. The CC-SLIQ algorithm inexpensively evaluates split points that are twice the cluster size *k* and results in a compact and accurate tree, scalable for large datasets as well as classified datasets with a large number of attributes, classes, and records. The classification accuracy of this technique has been compared to the existing SLIQ and Elegant decision tree methods on a large number of datasets from the UCI machine learning and Weather Underground repository. The experiments show that the proposed algorithm reduces the computation of split points by 95.62%, decision rules generated by 56.5% and also leads to better mean classification accuracy of 79.29%, thus making it a practical tool for data mining.**

***Index Terms*—Classification; *k*-means Clustering; Decision Tree; Gain ratio; SLIQ algorithm.**

I. Introduction

DECISION trees have been found to be the best algorithms for data classification, providing good accuracy for many problems in practical time. In a decision tree, each branch node represents the choice between a number of alternatives and each leaf node represents a classification decision [1, 2]. Decision trees are applied in decision analysis problems to help identify the most suitable strategy for reaching a concrete goal [3]. Decision trees provide classification rules and can be understood easily. However, the accuracy of decision trees has always been a problem. The ID3 decision tree algorithm was first proposed by Quinlan [4] and there have since been several enhancements to the original algorithm, including the C4.5 algorithm [5, 6]. Several approaches have been developed to improve the quality of decision tree results from different perspectives such as the C4.45 and C4.55 decision tree algorithms suggested by Mahmood et al. [7] provide promising and interesting results.

One of the main drawbacks of ID3 is the attribute selection measure used. The splitting attribute selection measure information gain in ID3 tends to favor attributes with a large number of distinct values. This drawback was overcome to some extent by the introduction of a new measure called gain ratio. Gain ratio takes into account the information about the classification gained by splitting on the basis of a particular attribute. The Gini index is an alternative proposal to gain ratio and is used in the SLIQ (Supervised Learning in Quest) algorithm by Manish et al. [8]. Several enhancements to ID3 have been proposed to overcome these drawbacks. These drawbacks are also dealt with in SLIQ.

Other algorithms that use the Gini index as a split measure are the SPRINT [9], CLOUDS [10], CMP-S [11], and Elegant decision tree [12]. The SPRINT algorithm aims to parallelize SLIQ. In CLOUDS, the range of each numeric attribute is divided into q intervals using a quantization technique. A major improvement of CMP-S over CLOUDS is that it scans the dataset only once. Various approaches have been suggested by Chandra et al. [13, 14], Huacheng Zhang et al. [15] for improving the efficiency of the SLIQ decision tree algorithm. In the Elegant decision tree algorithm, the Gini index is computed not only on every successor pair of attribute values but also over different attribute value ranges to determine the exact split point that improves the performance of SLIQ. Another approach has been to improve the performance of SLIQ is by computing the Gini index only at those points where the class information changes.

The drawback of the SLIQ algorithm is that a large number of Gini indices must be computed at each node of the decision tree. In order to decide which attribute is to be split at each node, the Gini indices must be computed for all attributes and for each successive pair of values for all patterns that have not yet been classified. SLIQ proposes to scan each attribute and each value corresponding to an attribute in order to ascertain the best split. However, in the case of large databases with numerous records, this process of scanning each attribute and each corresponding pair of successive values leads to a huge number of computations. Big datasets that consist of large groups of very similar instances with a few small groups of unusual cases can be a problem for this kind of classifier, as the classifiers tend to overlook corner cases. One approach to solving this problem is to classify the data into groups that share some features between them so that all groups get a similar number of instances, possibly decreasing the error rate.

This paper presents a novel method to improve the efficiency of classifier training by cascading the *k*-means clustering [24] and SLIQ decision tree learning methods. In particular, the method:

1) eliminates the presorting of data for each numeric attribute at the beginning of the tree growth phase,
2) eliminates the computation of a large number of Gini splitting indices for an attribute at a node,

Manuscript received November 25, 2013; revised August 18, 2014.

Narasimha Prasad L V is with the Department of Computer Science and Engineering, S V University College of Engineering, Hyderabad, India (e-mail: lvnprasad@yahoo.com).

Mannava Munirathnam Naidu is with the Department of Computer Science and Engineering, Gudlavalleru Engineering College, Gudlavalleru, India (e-mail: mmnaidu@yahoo.com).

3) results in smaller decision trees with fewer rules,
4) uses gain ratio instead of the Gini index, taking into account the information about the classification gained by splitting on the basis of a particular attribute.

The CC-SLIQ decision tree algorithm constructs the binary decision tree by cascading two machine learning algorithms: *k*-means clustering and SLIQ decision tree learning. The aim of clustering algorithms is to classify the data into groups that share some features between them. In the first stage, *k*-means clustering is performed on training instances to obtain '*k*' disjoint clusters. We choose *k*-means clustering because it is a data-driven method with relatively few assumptions on the distributions of the underlying data. In addition, its greedy search strategy guarantees at least a local minimum of the criterion function, thereby accelerating the convergence of clusters on large datasets. Each *k*-means cluster represents a region of instances that are similar in terms of Euclidean distances between the instances and their cluster centroids. In the second stage, the *k*-means method is cascaded with SLIQ decision tree learning by building an SLIQ decision tree using the instances in each of the *k*-means clusters.

We performed experiments on twelve benchmark datasets extracted from the UCI machine learning data repository [16] and nineteen real-world large datasets in the prediction of precipitation for various sub-continental regions from Weather Underground (http://www.wunderground.com). The performance of the CC-SLIQ algorithm was empirically compared with the SLIQ and Elegant decision tree classification algorithms as well as with other popular machine learning classifiers: naive Bayes and back propagation.

This paper is organized as follows. Section II describes the related work done to date. We briefly describe SLIQ decision tree in Section III. Section IV introduces the proposed CC-SLIQ decision tree algorithm. Section V provides the experimental results. A conclusion is drawn in Section VI.

II. Related Work

Various methodologies have been developed for constructing decision trees such as ID3 [4], C4.5 [5, 6], CART [17], CTC [18], and FDT [19]. Nevertheless, all these methodologies needs to hold the total training set in memory when constructing a decision tree. As a result, they cannot be applied to huge training sets.

Other methodologies have been developed for constructing decision trees from huge training sets, for instance, SLIQ [8], SPRINT [9], CLOUDS [10], Rain Forest [20] and BOAT [21]. However, all of these use lists for keeping a dataset in main memory; for each attribute in the dataset, these algorithms assign a list. The problem is that some of these lists require more space than the one required to store the entire training set. Other algorithms, like BOAT [21], ICE [22], and VFDT [23], are incremental algorithms. Both BOAT and ICE use a subset of training instances for constructing a decision tree; but for huge datasets, to search this subset of instances may be too expensive. In VFDT, the user needs to define values for three parameters before constructing a decision tree, which could be very difficult in practice.

The CC-SLIQ algorithm proposed here solves some of the limitations highlighted above. Our algorithm processes the entire training set without storing it in memory and the parameters can be easily defined by the user. In addition, our algorithm is faster than the most recent methodologies discussed in the literature for constructing decision trees from huge datasets.

III. SLIQ Decision Tree Algorithm

The SLIQ (Supervised Learning in Quest) was developed by the Quest team at the IBM Almaden Research Center to handle both numeric and categorical datasets. A tree is generated by splitting any one of the attributes at every level. The basic intent is to build the tree that is more accurate for prediction.

SLIQ employs pre-sorting of the training dataset T. For set T, a sorted attribute list is prepared along with its corresponding class label and the split points are evaluated using equation (1) whenever there is a change in the class label

$$split\ point = mid\ point(changed\ class\ labels) \quad (1)$$

and use equation (2) to compute the value of $G\_info(T)$ for the class label.

$$G\_info(T) = 1 - \sum_{i=1}^{N} P_i^2 \quad (2)$$

The notations used are given in Table I. A SLIQ decision tree classifier recursively divides the training set into partitions so that most or all of the records are of a similar class. The algorithm starts with the entire dataset at the root node. The dataset is partitioned by splitting the criteria into subsets according to the Gini index. The attribute containing the split point that maximizes the reduction in impurity or, equivalently, has the minimum Gini index, is used to split the node. The value of a split point depends upon how well it separates the classes. The splitting is carried out for the dataset by choosing the attribute that will best separate the remaining samples of the nodes apportioned into the individual classes.

SLIQ eliminates the need to sort the data at every node of decision tree, despite the training datasets are sorted only once for every numeric attribute at the beginning of the tree growth phase. CART [17] and C4.5 [5] classifiers grow trees

TABLE I: Notations used in generalized format.

Symbol	Notation
T	Training dataset
C	Number of class labels
N	Number of instances
n	Number of split points
p_i	The probability that a tuple in T belongs to class c_i
p_j	The probability that a tuple in T belongs to class c_j
$x_i^{(j)}$	Data point
c_j	Cluster centre
$\left\|x_i^{(j)} - c_j\right\|^2$	Distance measure between $x_i^{(j)}$ and c_j
sp	Split point value

depth-first and repeatedly sort the data at every node of the tree to arrive at the best splits for numeric attributes.

The *G_info* (v_q) value for each and every attribute is calculated according to the equation (3).

$$G\_info\,(v_q) = \sum_{j=1}^{n} P_j \left[1 - \sum_{i=1}^{N} P_i^2\right] \tag{3}$$

The *Gini index* (v_q) value is obtained using the equation (4) by subtracting $G\_info\,(v_q)$ from *G_info* (T).

$$Gini\ index\,(v_q) = G\_info\,(T) - G\_info\,(v_q) \tag{4}$$

The maximum $Gini\,index\,(v_q)$ value in equation (5) is considered the best split point and becomes the root node.

$$Best\ split\,point\ = maximum\,(Gini\,index\,(v_q)) \tag{5}$$

This procedure is repeated until every node ends with a unique class label.

IV. CC-SLIQ Decision Tree Algorithm

SLIQ is a decision tree classifier that can handle both numeric and categorical attributes. SLIQ uses a pre-sorting technique in the tree growth phase for numeric attributes, where sorting time is a dominant factor when finding the best split at a decision tree node. In the SLIQ algorithm, the Gini index is evaluated at every successive midpoint of the attribute values. However, the efficiency of the SLIQ algorithm was improved by evaluating the Gini index only at the mid points of the attributes where there is a change in class label information [14]. However, by doing so, the performance measures of the decision tree built does not change. Therefore, CC-SLIQ employs a scheme that does away with the need to sort the data at every node of the decision tree. Instead, the training data need to be partitioned using *k*-means clustering only once for each numeric attribute at the beginning of the tree growth phase. In addition, the split point value is computed at the cluster boundaries at both the beginning and end of the cluster segments. Consequently, splits of all the leaves of the current tree are simultaneously adopted in one pass over the data. The experimental implementation methodology of CC-SLIQ algorithm consists of four stages: 1) a *k*-means algorithm to group N data points into *k* disjoint clusters, where *k* is determined by an auto detection cluster classifier algorithm explained later in this section; 2) identification of the split points; 3) evaluation of the gain ratios for all the attributes; and 4) decision tree construction.

Function CC-SLIQ decision tree growth and split points

1) Read dataset (*T*) to select the root node of the CC-SLIQ decision tree.
2) Generate an attribute list for each attribute of *T*.
3) Compute $entropy\,(T)$ for each class label using the equation (6).

$$entropy\,(T)\ = -\sum_{i=1}^{N} p_i \log_2 p_i \tag{6}$$

4) Partition the training dataset along with the class label for each attribute 'v_q' using *k*-means clustering and mark the start and end value positions of each cluster segment as 's_p' where $p \leq 2k$, k is cluster size.
5) Create two subsets for each split point 's_p' such that left subset has values less than 's_p' and right subset has values greater than or equal to 's_p'.
6) Compute *Info*(v_q) for each and every attribute 'v_q' using the equation (7).

$$info\,(v_q) = \sum_{j=1}^{n} P_j \left[-\sum_{i=1}^{N} P_i \log_2 P_i\right] \tag{7}$$

Where P_j is the probability that a tuple in *T* belongs to class label c_j and P_i is the probability that a tuple in *T* belongs to class label c_i.

7) Calculate $Gain\,(v_q)$ for each and every attribute using the equation (8)

$$Gain\,(v_q)\ =\ entropy(T) - Info(v_q) \tag{8}$$

8) Compute $Split\ info\,(v_q)$ for each and every attribute using the equation (9)

$$Split\ info\,(v_q)\ = -\sum_{i=1}^{n} \frac{|T_i|}{|T|}\ \log_2\left(\frac{|T_i|}{|T|}\right) \tag{9}$$

9) The $Gain\ Ratio\,(v_q)$ in the equation (10) is the proportion of information generated by the split:

$$Gain\ Ratio\,(v_q)\ =\ \frac{Gain(v_q)}{Split\ info(v_q)} \tag{10}$$

10) The maximum $Gain\ Ratio\,(v_q)$ is considered to be the best split point and becomes the root node.

$$Best\ split\ point\ =\ maximum\,(Gain\ Ratio\,(v_q))$$

11) Repeat Steps 5 through 10, until there are no further partitions for the attributes.

k-means clustering algorithm is one among the simplest unsupervised learning methodologies that solves the well-known clustering problem [24]-[29]. *k*-means clustering causes problems when the *k* parameter in *k*-means is set to a value that is considerably less than the inherent number of natural groupings within the training data. The *k*-means procedure, when initialized with a low *k* value, underestimates the natural groups within the training data and, therefore, will not capture the overlapping groups within a cluster, forcing instances from different groups to be a part of the same cluster. The process follows an easy way to classify a given dataset choosing *k* cluster centers to coincide with *k* randomly chosen patterns or *k* randomly defined points inside the hypervolume containing the dataset. The major thought is to identify *k* centroids, for each cluster. This methodology aims to minimize an objective function, in this case the squared error function is given in equation (11).

$$J = \sum_{j=1}^{k} \sum_{i\,=\,1}^{n} \left\| x_i^{(j)} - c_j \right\|^2 \tag{11}$$

where $\left\| x_i^{(j)} - c_j \right\|^2$ is a chosen distance measure between a data point $x_i^{(j)}$ and the cluster center c_j, and *J* is an indicator of the distance of 'n' data points from their respective cluster centers. The methodology is composed of the following steps:

Function *k*-means clustering

1) Place *k* points into the object space. These points represent the initial group centroids.
2) Assign each object to the group that has the closest centroid.
3) When all objects have been assigned, recalculate the positions of the *k* centroids.
4) Repeat Steps 2 and 3 until the centroids no longer move. This creates a separation of objects into groups from which the metric to be minimized can be calculated.

A fundamental problem in *k*-means clustering is to determine the number of clusters, which is usually taken as prior or fixed. The selection of a good value for *k* can affect the overall accuracy of the algorithm, and clustering solutions may vary as different numbers of clusters are specified. A clustering technique would most possibly recover the underlying cluster structure, given a good estimate of the true number of clusters.

This paper uses a heuristic based optimum decision cluster algorithm that is designed to achieve this goal [30, 31]. Choosing a value for *k* by visual inspection can be automated by using the percentage of variance of clusters that determines the optimum number of clusters. This method finds the optimum number of clusters automatically, based on the relationship between consecutive differences among the data points [32]. The algorithm to compute the optimum number of clusters is as follows.

Function optimum decision cluster size

1) Read n records of an attribute and store in i^{th} row of jagged array (initially i = 0).
2) Compute the consecutive difference of i^{th} row and store in $(i+1)^{th}$ row.
3) Repeat step 2 until i $<$ n.
4) The highest row index which contains the maximum value from 1 to 9 (heuristic approach) determines the cluster size (k).

The training data is a set $T = \{x^1, x^2, ..., x^s\}$ of already classified data patterns. Each data pattern $x^s = ({x_1}^s, {x_2}^s, {x_3}^s, ..., {x_k}^s)$ represents the q^{th} attribute value of x^s where q is the total number of attributes. The training data is augmented with a vector $\psi = (C_1, C_2, ..., C_N)$, where C_i represents the class to which each data pattern belongs, and C is the total number of classes. Imagine selecting one data pattern from a set T of training data patterns randomly and announcing it belongs to the class C_i. This message has the probability $\mathrm{p_i} = |\mathrm{C_i}|/|\mathrm{T}|$, where $|\mathrm{C_i}|$ is the number of data patterns that belong to class C_i. The information contained in this announcement is $-\log_2(\mathrm{p_i})$. In general, if we are given a probability distribution $\mathrm{P} = (\mathrm{p_1}, \mathrm{p_2}, ..., \mathrm{p_n})$, using equation (6) we calculate $entropy\,(T)$ conveyed by this distribution.

When applied to the set T of training data pattern, the average amount of information needed is *Info(v_q)* to identify the class of a data pattern of set T. Consider a similar measurement after set T has been partitioned in accordance with the n subsets on the q^{th} attribute, v_q. The expected information requirement is obtained by evaluating equation (7), is the weighted sum over the subsets. The quantity, Gain(v_q) is then computed for each attribute list using equation (8). This epitomizes the difference between the information needed to identify an element of set T and the same information after the value of attribute v_q has been obtained, i.e., the gain in information due to attribute v_q. By analogy with equation (8), the split information can be obtained by using equation (9).

The gain ratio v_q defined in equation (10) is the proportion of information generated by the split that is useful for classification. The maximum gain ratio v_q is considered to be the best split point and becomes the root node.

At each node, the CC-SLIQ decision tree algorithm chooses the attribute of the data that most efficiently splits its set of data patterns into subsets rich in one class or the other. Its criterion is the gain ratio that results from choosing an attribute for splitting the data. Because the attribute 'V' has a continuous range, we can examine the values for this attribute in the training set. Then for each value V_q^j, where j = 1, 2, ..., n. We partition the records into the number of clusters *k*. For each of these *k* partitions we compute the *gain ratio* V_q^j and select a partition that maximizes it. The gain ratio is used to rank attributes and construct decision trees where every node is split conforming to the attribute with the greatest gain ratio among the attributes not yet chosen in the path from the root. Since all attributes are continuous, we obtain a binary decision tree.

The decision tree complexity is measured by the number of its terminal nodes. The computational complexity of the CC-SLIQ algorithm mainly lies in the exhaustive search for the best split value required at each node [33]-[39]. This computational complexity grows exponentially with respect to the size of dataset, and this is why the proposed *k*-means clustering technique helps. The main strength of the CC-SLIQ decision tree with the gain ratio attribute selection measure classifier is that it provides understanding and insight into the data. More insight can be obtained by using the information gain and Gini index attribute selection measures. Here, we briefly review them.

A. Information gain

Information gain is an impurity-based criterion that uses the entropy measure (originally from information theory) as the impurity measure. Taking the class label attribute and calculating the entropy, as shown in equation (6), the information gain of an attribute is calculated from equation (7). The gain of an attribute is calculated in order to choose the root node from equation (8). The attribute that has the maximum gain value is chosen for the root node.

B. Gini index

Gini index is an impurity-based criterion that measures the divergences between the probability distributions of the target attribute's values. The value of $G\_info\,(T)$ is calculated for the class label using equation (2). The value for *Gini index* (v_q) is calculated for each and every attribute using equation (3). The value is obtained by finding the difference between $G\_info\,(T)$ and $G\_info\,(v_q)$ using equation (4). The maximum *Gini index* (v_q) value is considered the best split point and is used for the root node.

In this paper, the CC-SLIQ algorithm constructs decision trees from a set of training data using the concepts of

information gain, Gini index, and gain ratio and comparing the results. The effectiveness of a CC-SLIQ classifier is in the formation of fewer uniform clusters, helping to reduce the number of split points at each node of the tree and, at the same time, increasing the classifier accuracy. The ideal goal is to produce compact, accurate and scalable trees in a short time.

C. Illustration

The steps in the proposed algorithm are explained in detail using a subset of the precipitation dataset shown in Table II. The dataset contains five attributes, namely humidity (A1), temperature (A2), atmospheric pressure (A3), wind speed (A4), dew point (A5), and a class label. The data are labeled into two classes: Class 1 (Z1) denotes rain and Class 2 (Z2) denotes no-rain. The function optimum decision cluster size determines the cluster size for the attributes. Each attribute value is clustered, along with the corresponding class label information. The cluster sizes obtained for each of the attributes A1, A2, A3, A4, A5 are 2, 2, 5, 2, and 2 respectively.

The split point value is computed at the cluster boundaries both at the beginning and end of the cluster segments. The results in Tables III-VII show the number of clusters formed and the split values evaluated. The decision tree generated using the CC-SLIQ algorithm is shown in Fig. 1.

The gain ratio evaluated for each of the split point values for all attributes are illustrated in Table VIII. The best

TABLE II: Sample dataset for illustration of CC-SLIQ.

S. No.	A1	A2	A3	A4	A5	Class
1	88	26	1004	13	21	Z2
2	92	27	1005	13	21	Z1
3	86	27	1007	11	21	Z1
4	82	27	1006	11	21	Z1
5	76	27	1007	14	19	Z2
6	79	27	1008	11	20	Z2
7	75	27	1008	13	20	Z2
8	84	27	1007	13	20	Z2
9	88	26	1006	11	21	Z1
10	86	25	1005	16	19	Z1
11	78	28	1006	13	21	Z2
12	79	27	1008	13	19	Z2
13	80	28	1008	8	20	Z2
14	84	29	1009	6	21	Z2
15	76	27	1009	6	22	Z1

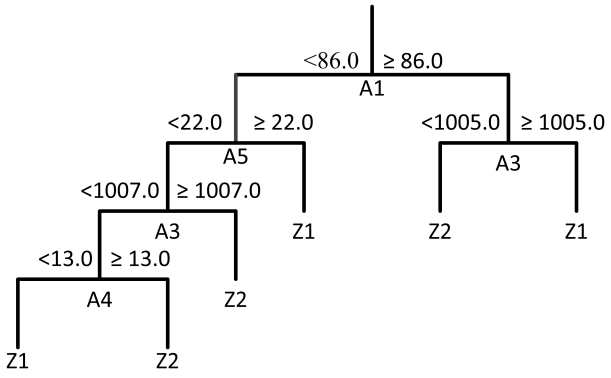


Fig. 1: CC-SLIQ decision tree.

TABLE III: Dataset segmented based on values of A1 attribute.

S. No.	A1	Class	Cluster number		Split point values
1	88	Z2	C1	→	88
2	92	Z1	C1		
3	86	Z1	C1		
4	82	Z1	C0	→	82
5	76	Z2	C0		
6	79	Z2	C0		
7	75	Z2	C0		
8	84	Z2	C0		
9	88	Z1	C1		
10	86	Z1	C1	→	86
11	78	Z2	C0		
12	79	Z2	C0		
13	80	Z2	C0		
14	84	Z2	C0		
15	76	Z1	C0	→	76

TABLE IV: Dataset segmented based on values of A2 attribute.

S. No.	A2	Class	Cluster number		Split point values
1	26	Z2	C0	→	26
2	27	Z1	C1	→	27
3	27	Z1	C1		
4	27	Z1	C1		
5	27	Z2	C1		
6	27	Z2	C1		
7	27	Z2	C1		
8	27	Z2	C1		
9	26	Z1	C0		
10	25	Z1	C0	→	25
11	28	Z2	C1		
12	27	Z2	C1		
13	28	Z2	C1		
14	29	Z2	C1		
15	27	Z1	C1	→	27

TABLE V: Dataset segmented based on values of A3 attribute.

S. No.	A2	Class	Cluster number		Split point values
1	1004	Z2	C0	→	1004
2	1005	Z1	C1	→	1005
3	1007	Z1	C2	→	1007
4	1006	Z1	C3	→	1006
5	1007	Z2	C2		
6	1008	Z2	C4	→	1008
7	1008	Z2	C4		
8	1007	Z2	C2	→	1007
9	1006	Z1	C3		
10	1005	Z1	C1	→	1005
11	1006	Z2	C3	→	1006
12	1008	Z2	C4		
13	1008	Z2	C4		
14	1009	Z2	C4		
15	1009	Z1	C4	→	1009

TABLE VI: Dataset segmented based on values of A4 attribute.

S. No.	A2	Class	Cluster number		Split point values
1	13	Z2	C0	→	13
2	13	Z1	C0		
3	11	Z1	C1	→	11
4	11	Z1	C1		
5	14	Z2	C0		
6	11	Z2	C1		
7	13	Z2	C0		
8	13	Z2	C0		
9	11	Z1	C1		
10	16	Z1	C0		
11	13	Z2	C0		
12	13	Z2	C0	→	13
13	8	Z2	C1		
14	6	Z2	C1	→	6
15	6	Z1	C1		

TABLE VII: Dataset segmented based on values of A5 attribute.

S. No.	A2	Class	Cluster number		Split point values
1	21	Z2	C0	→	21
2	21	Z1	C0		
3	21	Z1	C0		
4	21	Z1	C0		
5	19	Z2	C1	→	19
6	20	Z2	C0		
7	20	Z2	C0		
8	20	Z2	C0		
9	21	Z1	C0		
10	19	Z1	C0		
11	21	Z2	C0		
12	19	Z2	C1	→	19
13	20	Z2	C0		
14	21	Z2	C0		
15	22	Z1	C0	→	22

splitting attribute is the maximum among the gain ratios computed. Here, for attribute A1, a split point value of 86.0, has the maximum gain ratio and hence is chosen to be the root node. A similar process is repeated to compute the gain ratio values for various split points.

V. Experimental Results

We design two experiments to evaluate the performance of the CC-SLIQ decision tree algorithm. The first one is on twelve UCI machine learning repository benchmark datasets and second experiment is on nineteen datasets extracted from weather underground (http://www.wunderground.com) to predict the useful patterns for the hydro-metro prediction of precipitation. Prediction of precipitation is necessary as it has to be considered during the financial planning of a country. The meteorological departments of every nation are very keen in recording the datasets of precipitation which are huge in content. Hence, decision tree learning is one of the competitive tools which would extract the relation between the datasets and their attributes. It should be noted that all the test results shown in this section were simulated on an Intel Pentium i5 PC.

TABLE VIII: Gain ratio values for finding split points.

Iteration	A1		A2		A3		A4		A5	
	Split value	Gain ratio	Split value	Gain ratio	Split value	Gain ratio	Split value	Gain ratio	Split value	Gain ratio
Step1	**86.0**	**0.271**	26.0	0.149	1007.0	0.149	13.0	0.078	22.0	0.264
Step2	76.0	0.072	25.0	-1.0	1007.0	0.120	13.0	0.236	**22.0**	**0.573**
Step3	82.0	0.214	25.0	-1.0	**1007.0**	**0.367**	13.0	0.143	21.0	0.214
Step4	82.0	1.0	25.0	-1.0	1007.0	-1.0	**13.0**	**1.0**	22.0	-1.0
Step5	88.0	0.176	27.0	0.176	**1005.0**	**1.0**	13.0	0.176	21.0	0.1009

A. Experimental results on UCI datasets

The performance of the CC-SLIQ decision tree algorithm was compared with that of the SLIQ and Elegant decision trees using twelve UCI machine learning repository benchmark datasets. The datasets represent a wide range of domains and data characteristics. The datasets, in ascending order of size, are shown in Table IX.

The effectiveness of selecting the best split attribute is defined by the reduction of impurity from the parent to the child nodes before splitting. The larger the reduction of impurity, the better the selected split attributes. Hence, inorder to study the effectiveness of the CC-SLIQ, information gain, Gini index and gain ratio attribute selection measures were implemented and the results are tabulated in Table X. The CC-SLIQ with gain ratio measure has better mean classification accuracy over the information gain and Gini index measures.

The mean classification accuracy of the proposed CC-SLIQ is around 77.11% obtained via 10 runs of 10-fold stratified cross validation; while the mean classification accuracies using the SLIQ and Elegant decision trees are 74.72% and 73.36%, respectively, as shown in Table XI. The performance of CC-SLIQ outperforms both SLIQ and Elegant decision trees for most of the datasets except for Ionosphere and Breast cancer, where the accuracies are marginally less.

We also compared the performance of CC-SLIQ with other popular machine learning classifiers: naive Bayes and back propagation. The naive Bayes is a well-known parametric classification algorithm with very low time complexity and considerably good accuracy as well. However, it is based on an unrealistic assumption that: all attributes are independent given in the class. The back propagation model of neural networks has long training periods, normally learns with non-local information, often fails to converge, and at best converges to local error minima. Both methodologies fail to provide inference rules and draw conclusions with uncertain evidence. The experimental results shown in Table XII, prove that the CC-SLIQ method is competitive in terms of classification accuracy with naive Bayes and back propagation for most of the datasets.

The number of split points computed and rules evaluated during the construction of decision trees are depicted in Tables XIII and XIV respectively. Table XIII shows that CC-SLIQ greatly reduces the number of split points generated thus reducing the number of computations at every node taking low running time. In addition, there is a significant reduction of 83.29% over SLIQ decision tree and 70.91% over Elegant decision trees in computing the number of split points. A graph plotted between the number of instances and the split points generated by SLIQ and CC-SLIQ decision trees, is shown in Fig. 2(a).

Consequently, Table XIV shows that the CC-SLIQ algorithm generates fewer rules, a reduction of 17.27% compared

TABLE IX: Datasets used for the system evaluation.

Dataset	Instances	Attributes	Classes
Zoo [Richard S. Forsyth, 1990]	101	16	7
Iris [R.A. Fisher, 1988]	150	4	3
Heart[David W. Aha, 1988]	267	22	2
Haberman [Tjen-Sien Lim, 1999]	290	3	2
Liver [Richard S. Forsyth, 1990]	346	6	2
Ionosphere [Vince Sigillito, 1989]	422	34	2
Diabetes [Michael Kahn, 1994]	517	8	2
Breast cancer [Ming Tan, 1988]	699	9	2
Vehicle [Dr.Pete Mowforth, 1987]	846	18	4
Yeast [Kenta Nakai, 1996]	1484	8	9
Abalone [Sam Waugh, 1995]	4117	8	3
Letter [David J. Slate, 1991]	20000	16	26

TABLE X: Comparison of CC-SLIQ gain ratio classifier accuracies with entropy and Gini index measures.

Dataset	Instances	CC-SLIQ accuracy % using		
		Entropy	Gini Index	Gain ratio
Zoo	101	88.01	88.01	90.01
Iris	150	75.01	75.01	75.01
Heart	267	66.00	67.50	72.51
Haberman	290	77.77	77.77	77.77
Liver	346	67.81	67.81	68.96
Ionosphere	422	72.51	87.69	92.31
Diabetes	517	60.01	68.01	80.00
Breast cancer	699	90.62	90.62	93.75
Vehicle	846	66.09	67.24	71.26
Yeast	1484	48.78	47.43	49.86
Abalone	4117	55.61	54.08	54.59
Letter	20000	78.17	79.02	99.35
Mean		70.53	72.51	77.11

TABLE XI: Classification accuracy of CC-SLIQ with other decision trees on UCI datasets.

Dataset	Accuracy (%)		
	SLIQ	Elegant	CC-SLIQ
Zoo	88.01	88.01	90.01
Iris	91.66	91.66	75.01
Heart	70.51	62.51	72.51
Haberman	75.92	70.37	77.77
Liver	66.66	60.91	68.96
Ionosphere	98.46	92.31	92.31
Diabetes	72.01	79.42	80.00
Breast cancer	96.80	96.87	93.75
Vehicle	54.59	57.49	71.26
Yeast	48.78	47.43	49.86
Abalone	55.61	50.51	54.59
Letter	77.67	83.75	99.35
Mean	74.72	73.36	**77.11**

TABLE XII: Classification accuracy of CC-SLIQ with other classifiers on UCI datasets.

Dataset	Accuracy (%)		
	naive Bayes	Back propagation	CC-SLIQ
Zoo	81.82	83.12	90.01
Iris	88.88	94.44	75.01
Heart	54.31	55.84	72.51
Haberman	71.34	73.92	77.77
Liver	56.31	55.86	68.96
Ionosphere	78.44	84.48	92.31
Diabetes	61.36	68.18	80.00
Breast cancer	86.66	85.45	93.75
Vehicle	43.91	76.52	71.26
Yeast	61.02	58.77	49.86
Abalone	50.87	56.74	54.59
Letter	62.24	80.14	99.35
Mean	66.43	72.78	**77.11**

TABLE XIII: Comparison of split points generated in CC-SLIQ with other decision trees on UCI datasets.

Dataset	Split points computed			% reduction in split points over	
	SLIQ	Elegant	CC-SLIQ	SLIQ	Elegant
Zoo	73	48	**25**	65.75	47.92
Iris	57	54	**20**	64.91	62.96
Heart	64	61	**31**	51.56	49.18
Haberman	81	32	**14**	82.72	56.25
Liver	333	124	**30**	90.99	75.81
Ionosphere	1682	1682	**107**	93.64	93.64
Diabetes	208	102	**31**	85.10	69.61
Breast cancer	67	27	**18**	73.13	33.33
Vehicle	2512	344	**68**	97.29	80.23
Yeast	1336	623	**32**	97.60	94.86
Abalone	1710	1698	**32**	98.13	98.12
Letter	4021	483	**53**	98.68	89.03
Mean	4021	439.83	**38.41**	**83.29**	70.91

TABLE XIV: Comparison of rules generated in CC-SLIQ with other decision trees on UCI datasets.

Dataset	SLIQ	Elegant	CC-SLIQ	% of reduction in rules over	
				SLIQ	Elegant
Zoo	11	11	11	0	0
Iris	11	9	8	27.27	11.11
Heart	74	76	73	1.35	3.94
Haberman	75	76	28	62.66	63.15
Liver	99	112	97	2.02	13.39
Ionosphere	28	29	24	14.28	17.24
Diabetes	122	105	101	17.21	3.80
Breast cancer	44	46	44	0	4.34
Vehicle	267	246	205	23.22	16.66
Yeast	489	446	426	12.88	4.48
Abalone	1336	1275	893	33.15	29.96
Letter	3041	2884	2636	13.31	8.59
Mean	466.41	442.91	378.8	**17.27**	14.72

to SLIQ and 14.72% compared to Elegant decision trees. This performance improvement is shown in Fig. 2(b). Hence, CC-SLIQ decision tree classifier have the ability to manage complex problems by offering an understandable representation, i.e., they are easier to interpret as they produce logical classification rules. This demonstrates how clustering techniques not only help to reduce the computational complexity of the SLIQ algorithm but also handle noisy environments and corner cases.

B. Experimental results on precipitation datasets

We conducted the second experiment on nineteen precipitation datasets with two class labels to predict how the present state of environmental variables humidity, temperature, pressure, wind speed and dew point affects the precipitation for rain/no-rain detection. Table XV shows the number of instances and average accuracies of CC-SLIQ over SLIQ and Elegant decision trees obtained via 10-fold stratified cross validation.

CC-SLIQ performs better than SLIQ and Elegant decision trees and the mean classification accuracy of CC-SLIQ is 80.67%, higher than SLIQ's 73.43% and Elegant's 73.45%. In addition, Table XVI shows that CC-SLIQ greatly reduces the number of split points, by 94.22% over SLIQ and 90.86% over Elegant decision trees. The values obtained are given in

TABLE XV: Classification accuracy of CC-SLIQ with other decision trees on precipitation datasets.

Dataset	Instances	Accuracy (%)		
		SLIQ	Elegant	CC-SLIQ
Zimbambwe	1319	79.71	80.23	83.16
Cairo	1535	95.33	94.68	97.13
Taiwan	1666	69.8	68.96	73.87
Botswana	1678	83.61	81.46	86.68
St.Petersberg	2148	72.9	73.23	72.98
Perth	2182	79.37	78.51	82.65
Adelaide	2183	72.19	71.87	77.66
Tokyo	2189	71.31	71.91	75.69
Jakartha	2529	74.06	77.12	91.66
Phillipines	4000	64.34	66.78	72.29
Lahore	4886	79.1	80.67	85.58
Washington	5507	26.33	22.78	68.59
Delhi	6015	82.65	81.42	87.22
Victoria	6044	69.68	70.41	74.2
Manama	6072	88	87.16	91.36
Brazil	6367	71.9	73.34	76.36
Bangkok	6659	71.29	70.33	77.52
Dallas	19659	67.9	66.86	73.82
San Francisco	24368	75.8	77.89	84.48
Mean		73.43	73.45	**80.67**

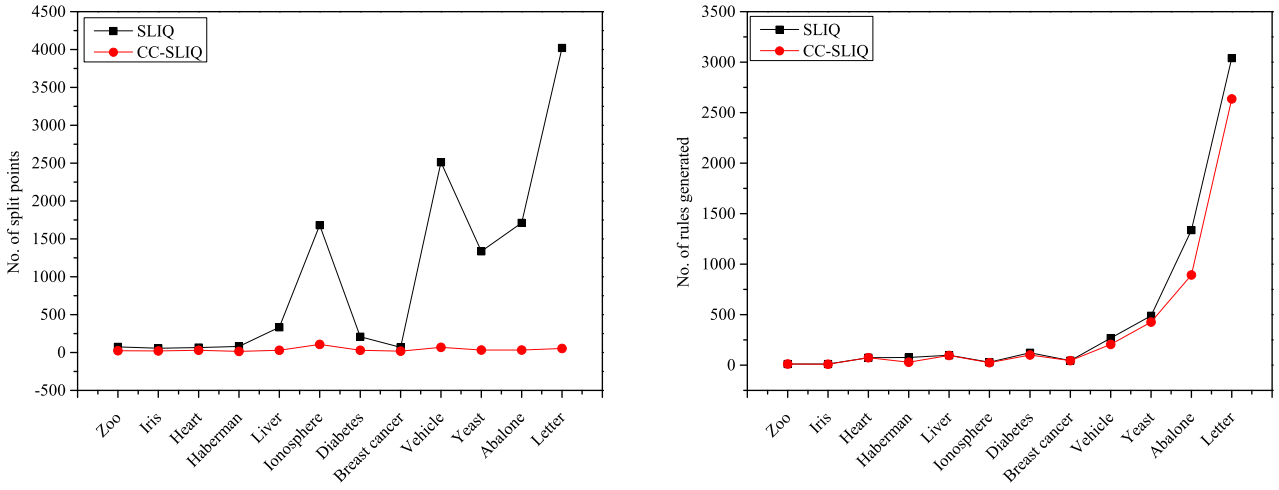
Fig. 3(a). It is not a surprise that CC-SLIQ performs better than naive Bayes, with a average accuracy of 80.67% vs. the 76.99% for naive Bayes and 80.68% for back propagation, shown in Table XVII.

CC-SLIQ, in-terms of rules generated on precipitation datasets is shown in Table XVIII. There is a reduction of 58.77% and 58.91% over SLIQ and Elegant decision tress respectively. This reduction is depicted in Fig. 3(b). The size of the decision tree constructed using CC-SLIQ is significantly smaller and still enjoys higher classification accuracy, handling both noisy environments and corner cases.

The size of a decision tree constructed using CC-SLIQ is significantly smaller than one constructed using SLIQ and Elegant decision tree methods. The comparison results show

TABLE XVI: Comparison of split points generated in CC-SLIQ with other decision trees on precipitation datasets.

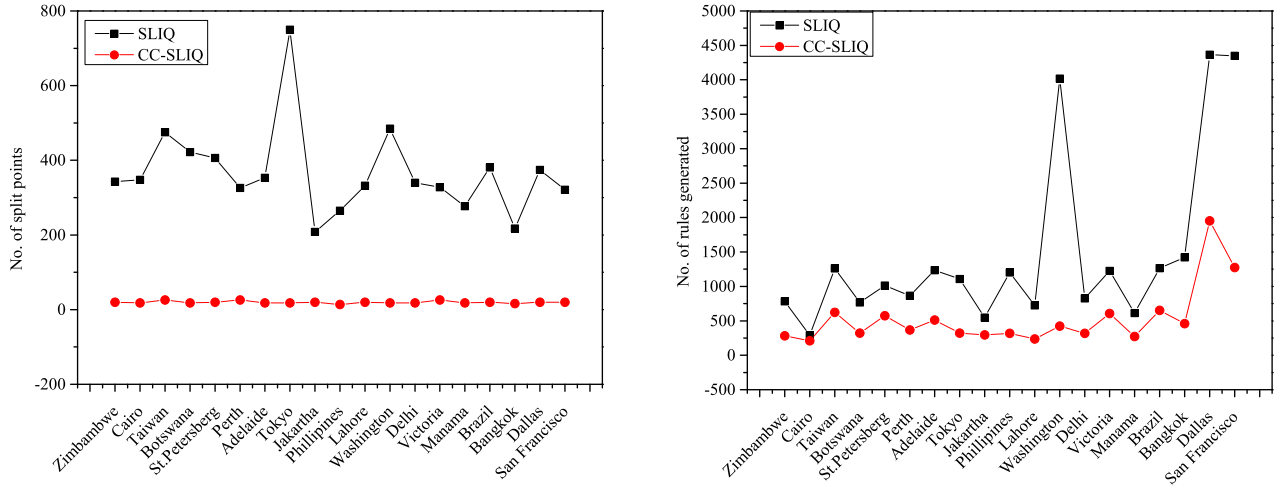
Dataset	Split points computed			% reduction in split points over	
	SLIQ	Elegant	CC-SLIQ	SLIQ	Elegant
Zimbambwe	343	191	**20**	94.17	89.53
Cairo	348	168	**18**	94.83	89.29
Taiwan	475	405	**26**	94.53	93.58
Botswana	422	255	**18**	95.73	92.94
St.Petersberg	406	250	**20**	95.07	92.00
Perth	326	200	**26**	92.02	87.00
Adelaide	353	236	**18**	94.90	92.37
Tokyo	750	260	**18**	97.60	93.08
Jakartha	208	147	**20**	90.38	86.39
Phillipines	265	214	**14**	94.72	93.46
Lahore	332	188	**20**	93.98	89.36
Washington	485	360	**18**	96.29	95.00
Delhi	340	206	**18**	94.71	91.26
Victoria	328	241	**26**	92.07	89.21
Manama	277	132	**18**	93.50	86.36
Brazil	381	269	**20**	94.75	92.57
Bangkok	217	141	**16**	92.63	88.65
Dallas	374	272	**20**	94.65	92.65
San Francisco	321	241	**20**	93.77	91.70
Mean	365.84	230.31	**19.68**	**94.22**	90.86



(a) Number of split points computed.

(b) Number of rules generated.

Fig. 2: Evaluation of CC-SLIQ on UCI datasets.



(a) Number of split points computed.

(b) Number of rules generated.

Fig. 3: Evaluation of CC-SLIQ on precipitation datasets.

that the CC-SLIQ not only performs better than SLIQ and Elegant decision trees in terms of mean classification accuracy but also takes less training time due to the improvement achieved in the competition of split points. Thus, CC-SLIQ is a very successful classification method.

VI. Conclusion

In this paper, we proposed the CC-SLIQ decision tree algorithm, a performance enhancement to the SLIQ decision tree presented by Manish et al. [8]. CC-SLIQ is found to be efficient in three important measures; (1) number of computations at each node, (2) decision rules generated, and (3) overall classification accuracy. In addition, the proposed CC-SLIQ approach results in decision trees that have smaller sizes and fewer rules than trees built with standard methods. This method is very useful in noisy environments and corner cases, a major weakness of SLIQ decision trees.

We have presented experimental results on UCI machine learning and precipitation repository datasets. The performance of the CC-SLIQ algorithm is compared with crisp SLIQ and other well known classification methods. The comparison results shows that CC-SLIQ has advantages in classifying large, high dimensional data with multiple classes and continuous attributes.

Consequently, it can serve as an efficient classifier for time applications. Currently, *k*-means clustering is used to partition training instances by selecting one attribute at a time. In future work, we plan to investigate allowing a partition that involves all the attributes at one time, significantly improves the decision tree classification performance. New multilayer clustering methods could be used for clustering complicated datasets as they are not only efficient to calculate the number of clusters, but could also improve the flexibility and effectiveness of the existing one layer clustering methods.

References

[1] Quinlan, J.R., "Simplifying Decision Trees," *International Journal of Man Machine Studies*, 1986, pp. 221−248.

TABLE XVII: Classification accuracy of CC-SLIQ with other classifiers on precipitation datasets

Dataset	Accuracy (%)		
	naive Bayes	Back propagation	CC-SLIQ
Zimbambwe	79.20	86.30	83.16
Cairo	94.02	97.01	97.13
Taiwan	66.45	75.13	73.87
Botswana	74.43	86.89	86.68
St.Petersberg	80.29	80.08	72.98
Perth	76.96	84.69	82.65
Adelaide	78.76	82.31	77.66
Tokyo	64.83	74.77	75.69
Jakartha	78.62	80.79	91.66
Phillipines	75.29	73.66	72.29
Lahore	83.97	79.73	85.58
Washington	58.95	73.84	68.59
Delhi	83.56	84.01	87.22
Victoria	70.36	69.90	74.20
Manama	89.86	89.10	91.36
Brazil	78.06	79.36	76.36
Bangkok	73.15	81.92	77.52
Dallas	73.75	73.79	73.82
San Francisco	82.45	79.66	84.48
Mean	76.99	80.68	**80.67**

TABLE XVIII: Comparison of rules generated in CC-SLIQ with other decision trees on precipitation datasets

Dataset	Rules generated			% of reduction in rules over	
	SLIQ	Elegant	CC-SLIQ	SLIQ	Elegant
Zimbambwe	784	797	281	64.15	65.81
Cairo	288	299	210	27.08	29.76
Taiwan	1260	1211	622	50.63	48.63
Botswana	768	748	323	57.94	56.81
St.Petersberg	1013	995	573	43.43	42.41
Perth	865	878	366	57.68	58.31
Adelaide	1234	1206	510	58.67	57.71
Tokyo	1106	1212	323	70.79	73.34
Jakartha	546	565	294	46.15	47.96
Phillipines	1204	1200	318	73.58	73.5
Lahore	723	719	237	67.21	67.03
Washington	4018	3818	423	89.47	88.92
Delhi	825	810	317	61.57	60.86
Victoria	1225	1204	605	50.61	49.75
Manama	614	634	271	55.86	57.25
Brazil	1264	1256	652	48.41	48.08
Bangkok	1422	1418	460	67.65	67.55
Dallas	4364	4344	1953	55.24	55.04
San Francisco	4347	4343	1274	70.69	70.66
Mean	1466.84	1455.63	**5526.94**	**58.77**	58.91

[2] Safavian, S.R. and Landgrebe, D., "A Survey of Decision Tree Classifier Methodology," *IEEE Transactions on Systems, Man and Cybernetics*, 1991, pp. 660−674.

[3] Quinlan, J.R., "Decision Trees and Decision Making," *IEEE Transactions on Systems, Man and Cybernetics*, 1990, pp. 339−346.

[4] Quinlan, J.R., "Introduction of Decision Trees," *Machine Learning*, 1986, pp. 81−106.

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

[6] Quinlan, J.R., "Improved use of Continuous Attributes in C 4.5," *Journal of Artificial Intelligence Research*, 1996, pp. 77−90.

[7] Mahmood, A., Rao, K.M. and Reddi, K., "A Novel Algorithm for Scaling up the Accuracy of Decision Trees," *International Journal of Computer Science and Engineering*, 2010, pp. 126−131.

[8] Manish, M., Agrawal, R. and Rissanen, J., "SLIQ: A Fast Scalable Classifier for Data Mining," *EDBT Proceedings of the International Conference on Extending Database Technology: Advances in Database Technology*, Springer-Verlag, 1996, pp. 18−32.

[9] Shafer, J., Rakeeh, A. and Manish, M., "SPRINT: A Scalable Parallel Classifier for Data Mining," *Proceedings of VLDB Conference*, 1996, pp. 544−555.

[10] Khaled, A., Sanjay, R. and Vineet, S., "CLOUDS: A Decision Tree Classifier for Large Datasets," *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, 1998, pp. 1−34.

[11] Haixun, W. and Carlo, Z., "CMP: A Fast Decision Tree Classifier Using Multivariate Predictions," *Proceedings of the International Conference on Data Engineering*, 2000, pp. 449−460.

[12] Chandra, B., Mazumdar, S., Vincent, A. and Parimi, N., "Elegant Decision Tree Algorithm for Classification in Data Mining," *Proceedings of the Third International Conference on Web Information Systems Engineering*, 2002, pp. 160−169.

[13] Chandra, B. and Varghese, P.P., "Fuzzy SLIQ Decision Tree Algorithm," *IEEE Transactions on Systems, Man and Cybernetics*, 2008, pp. 1294−1301.

[14] Chandra, B. and Varghese, P.P., "On improving efficiency of SLIQ decision tree algorithm," *Proceedings of the IEEE International Joint Conference on Neural Networks*, 2007, pp. 66−71.

[15] Huacheng, Z. and Wu, Xie., "Improvement of SLIQ Algorithm and its Application in Evaluation," *Proceedings of the IEEE International Conference on Genetic and Evolutionary Computing*, 2009, pp. 77−80.

[16] Merz, C.J. and Murphy, P.M., "In UCI Repository of machine learning databases," http://www.ics.uci.edu/ mlearn/MLRepository.html, Irvine, CA: University of San Francisco, Department of Information and Computer Science, 1996.

[17] Breiman, L., Friedman, J., Olshen, R. and Stone, C., "Classification and Regression Trees," 1^{st} ed., *Wadsworth International Group*, 1984.

[18] Perez, J.M., Muguerza, J., Arbelaitz, O. and Gurrutxaga, I., "Consolidated Tree Construction Algorithm: Structurally Steady Trees," *Proceedings of the International Conference on Enterprise Information Systems*, 2004, pp. 14−21.

[19] Janikow, C.Z., "Fuzzy Decision Trees: Issues and Methods," *IEEE Transactions on Systems, Man, and Cybernetics*, 1998, pp. 1−14.

[20] Johannes, G., Raghu, R. and Venkatesh, G., "RainForest−A Framework for Fast Decision Tree Construction of Large Datasets," *Data Mining and Knowledge Discovery*, Vol. 4, 2000, pp. 127−162.

[21] Johannes, G., Raghu, R., Venkatesh, G. and Wei, Y., "BOAT-Optimistic decision tree Construction," *ACM SIGMOD International Conference on Management of Data*, 1999, pp. 169−180.

[22] Hakil, Y., Khaled, A. and Sanjay, R., "Tree Based Incremental Classification for Large Datasets," CISE Department, University of Florida, 1999, pp. 1−23.

[23] Hang, Y., Simon, F., Guangmin, S. and Raymond, W., "A Very Fast Decision Tree Algorithm for -Time Data Mining of Imperfect Data Streams in a Distributed Wireless Sensor Network," *International Journal of Distributed Sensor Networks*, 2012, pp. 1−16.

[24] Hartigan, J. and Wong, M., "Algorithm AS136: A K-Means Clustering Algorithm," *Journal of the Royal Statistical Society*, 1979, pp. 100−108.

[25] Milligan, G. and Cooper, M., "An Examination of Procedures for Determining the Number of Clusters in a Dataset," *Psychometrika*, 1985, pp. 159−179.

[26] Milligan, G.W., Soon, S.C. and Sokol, M., "The Effect of Cluster Size, Dimensionality and the Number of Clusters on Recovery of True Cluster Structure," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1983, pp. 40−47.

[27] Guralnik, V. and Karypis, G., "A Scalable Algorithm for Clustering Sequential Data," *Proceedings of the IEEE International Conference on Data Mining Series*, 2001, pp. 179−186.

[28] Jarvis, R.A. and Patrick, E.A., "Clustering Using a Similarity Measure Based on Shared Nearest Neighbors," *IEEE Transactions on Computers*, 1973, pp. 1025−1034.

[29] Modha, D. and Spangler, W., "Feature Weighting in K-Means Clustering," *Machine Learning*, Springer, 2002, pp. 217−237

[30] Samir Tout, Junping Sun, and William Sverdlik, "A Hybrid Approach to Cluster Detection", *IAENG International Journal of Computer Science*, 34:1, 2007, pp. 105−110.

[31] Li-Yeh Chuang, Yu-Da Lin, and Cheng-Hong Yang, "Data Clustering Using Chaotic Particle Swarm Optimization", *IAENG International Journal of Computer Science*, 39:2, 2012, pp. 208−213.

[32] Vincent Granville, "Developing Analytic Talent: Becoming a Data Scientist," Indianapolis: John Wiley & Sons, Inc, 2014, pp. 141−143.

[33] Yan, Li., Edward, H., Korris, C. and Joshua, H., "Building a Decision Cluster Classification Model by a Clustering Algorithm to Classify Large High Dimensional Data with Multiple Classes," *Advances in Artificial Intelligence, Springer*, 2008, pp. 337−347.

[34] Patrice, B. and Marc, E., "Clustering by Means of Unsupervised Decision Trees or Hierarchical and K-means like Algorithm," *RIAOO Conference Proceedings*, 2000, pp. 344−363.

[35] Heena, S. and Navdeep, K.K., "Data Mining with Improved and Efficient Mechanism in Clustering Analysis and Decision Tree as a

Hybrid Approach," *International Journal of Innovative Technology and Exploring Engineering*, 2013, pp. 58−60.
[36] Liu, B., Xia, Y. and Yu, P.S., "Clustering Through Decision Tree Construction," *Proceedings of the International Conference on Information and Knowledge Management*, 2000, pp. 20−29.
[37] Ali, S.A., Sulaiman, N., Mustapha, A. and Mustapha, N., "K-Means Clustering to Improve the Accuracy of Decision Tree Response Classification," *Information Technology Journal*, 2009, pp. 1256−1262.
[38] Indrajit Saha, and Anirban Mukhopadhyay, "Improved Crisp and Fuzzy Clustering Techniques for Categorical Data", *IAENG International Journal of Computer Science*, 35:4, 2008, pp. 438−450.
[39] Barak, A. and Gelbard, R., "Classification by Clustering Decision Tree like Classifier Based on Adjusted Clusters," *Expert Systems with Applications*, 2011, pp. 8220−8228.