A Case Based Study on Decision Tree Induction with AVL-Tree

Suneetha Manne
Assistant Professor, Department of IT
V. R. Siddhartha Engineering College
Vijayawada ,Andhra Pradesh, INDIA
suneethamanne74@gmail.com

Suhasini Sodagudi
Assistant Professor, Department of IT
V. R. Siddhartha Engineering College
Vijayawada ,Andhra Pradesh, INDIA
ssuhasini09@gmail.com

Sita Kumari Kotha
Assistant Professor, Department of IT
V. R. Siddhartha Engineering College
Vijayawada ,Andhra Pradesh, INDIA
sitakumari.kotha@gmail.com

Abstract— **Data Mining is one of the eminent research fields to find interesting trends or patterns in large datasets. In order to work with the numeric or categorical data, the classification technique suits well for analyzing and processing for wider variety of large databases. Efficiency and scalability are fundamental issues concerning data mining in large databases. This paper describes the stability and quality of quantifiable elements among the given datasets and presented the representations of AVL trees for stability and Decision trees for quality. Further, the approach was extended and implemented for the employability of the students as large data sets for an educational institute.**

Keywords— **Classification, Decision tree induction, Attribute Oriented Induction, AVL tree**

I. INTRODUCTION

Computational efficiency and scalability are two important and challenging issues in data mining. Data mining is the automated discovery of nontrivial, previously unknown, and potentially useful patterns embedded in databases [1]. The increasing computerization of all aspects of life has led to the storage of massive amounts of data. Large scale data mining applications involving complex decision making can access billions of bytes of data. Hence, the efficiency of such applications is paramount.

Classification is a key data mining technique whereby database tuple acting as training samples, are analyzed in order to produce a model of the given data [2][18]. Each tuple is assumed to belong to a predefined class, as determined by one of the attributes, called the classifying attribute. Once derived, the classification model can be used to categorize future data samples, as well as provide a better understanding of the database contents. Classification has numerous applications including credit approval, product marketing, and medical diagnosis.

A number of classification techniques from the statistics and machine learning communities have been proposed [2][6][7]. A well accepted method of classification is the induction of decision trees [4][6]. A decision tree is a flowchart like structure consisting of internal nodes, leaf nodes, and branches. Each internal node represents a decision, or test, on a data attribute, and each outgoing branch corresponds to a possible outcome of the test. Each leaf node represents a class. In order to classify an unlabeled sample against the decision tree, a path is traced from the root to a leaf node which holds the class predicate for that sample. Decision trees can easily be converted into IFTHEN rules [6] and used for decision making.

The efficiency of the existing decision tree algorithms, such as ID3[4], C4.5[6] and CART[6], has been established for relatively small data sets[8]. Most decision tree algorithms have restriction that the training tuples should reside in main memory. In data mining applications, very large training sets of millions of examples are common. Hence, this restriction limits the scalability of such algorithms, where the decision tree construction can become inefficient due to swapping of the training samples in and out of main and cache memories.

The induction of decision trees from very large training sets has been previously addressed by the SLIQ [9] and SPRINT [10] decision tree algorithms. These proposed presorting techniques on disk resident data sets are too large to fit in memory. While SLIQ's scalability, however, is limited by the use of a memory resident data structure, SPRINT removes all memory restrictions and hence can handle data sets that are too large for SLIQ [10]. Unlike SLIQ and SPRINT, which operate on the raw low level data, we address the efficiency and scalability issues by proposing a different approach, consisting of three steps: *1) attribute oriented induction* [11][12], where concept hierarchies are used to generalize low level data to higher level concepts, *2) relevance analysis* [13], and *3) multilevel mining*[17], whereby decision trees can be induced at different levels of abstraction.

In the aspect of computer science, an AVL tree is the first data structure which was invented as a self-balancing binary search tree. Such trees are used to sustain the stability of quantifiable distinguishable elements in a given large dataset [14]. All the operations including search, insert, and delete take O(log n) time in both the average and worst cases, where n is the number of nodes in the tree prior to the operation. In insert and delete operations we observe that trees become unbalanced and in such cases the tree is required to be rebalanced by one or more rotations.

A binary tree is an AVL-tree if each node satisfies the BST property i.e root node is greater than left node and lesser than the right node and the difference between the heights of the sub tree should not exceed one. The balance factor of a node in a AVL tree is the height of its left sub tree minus the height of its right sub tree (sometimes opposite) and a node with balance factor 1, 0, or -1 is considered balanced. A node with any other balance factor is considered unbalanced and requires rebalancing the tree. The balance factor is either stored directly at each node or can be computed from the heights of the sub trees. AVL trees are more statically balanced, so these are faster for lookup intensive applications.

Statistically, this paper is worked on an educational enterprise to deal the huge dataset and applying the classification model by attribute oriented induction method. The student data is generalized into various categories and removed irrelevant attributes to obtain a concise set of data and then at multiple levels data is mined and finally succeeded in delivering class labels to test the nodes in decision tree in handling few decisions for instance whether a given student is eligible for placement or not. The combination of mathematics and computational capabilities generates the tree which helps in generalization of data. Hence, decision trees forms the basis in data mining in extracting the attributes from databases and correlated to the problem specific in automating a decision making system. Such trees rate high in information extraction and prediction which can be achieved by data preprocessing.

The approach is specified as the following sessions. Session II presents the Related Work, Session III presents the Classification using Decision Tree Induction, Session IV specifies the Decision Tree Construction using the proposed approach and Session V illustrates the proposed Decision Tree method with AVL trees. We conclude our study in Session VI and discuss the possible extensions based on our current work.

II. RELATED WORK

Binary tree is a tree which has only 2 branches and these branches can in turn have 2 sub branches and so on. Tree structures perform all the basic operations like insert, update, delete and search can be done with time proportionality as a constant to the height of the tree. The tree of short height is easily understandable and we need to maintain the height as a running component. To ensure this red-black trees, AVL trees must be used. B-trees are balanced trees that try to minimize the number of disk accesses. B-trees are good for searches in linear manner, but cause some overhead issues in wasting space. B-trees have to be rebuilt after a crash and so these are not more valuable

Regression tree is quite similar to a B-tree used for indexing multidimensional information [4]. Here in this approach, a node is allowed to find its way downward the tree to the appropriate location. Each entry within a non leaf node stores two pieces of information: about its child node and its bounding box. Each entry within a leaf node stores two pieces of information: actual data element and its bounding box. In insertion and deletion algorithms, the bounding boxes from the nodes will ensure that closer elements are placed in the same leaf node. Similarly, the searching algorithms (e.g., intersection, containment, nearest) use the bounding boxes to decide whether or not to search inside a child node. Appropriately, most of the nodes in the tree are never touched and used during a search. In this scenario, just like B-trees, R-trees are viewed suitable for databases, where nodes can be paged to memory when needed. Likewise, node insertion and deletion are difficult here.

AVL trees are performed exactly as unbalanced binary search tree [15]. In a look up process of a node, once a node has been found in a balanced tree, the next or previous nodes can be explored by the reduction of the value of an asset by prorating its cost in a series of time. Few cases require traversing up to 2×log(*n*) links where n is the number of nodes in the tree. Exploring all n nodes in this manner will use each link exactly twice, and there will be (n-1) links, at the rate of 2 ×(n-1)/n series of time, approximately 2.

Decision tree learning is a method in data mining that uses decision tree as a predictive model and uses the tree to create a model that predicts the value of a target variable based on several input variables. The leaves represent the classifications and the branches represent the conjunctions that lead to different classifications. The meta data that is stored in non leaf node affects in sorting of the tree. For every possible value of the input variable, each non leaf node represents one of the input variables and it is represented as an edge to children. For the given values of the input variables, each leaf node denotes a value of the target variable and it is represented by the path from the root to the leaf. A tree can be understood by generalizing the source set into subsets based on an attribute value test. This process is repeated on each derived subset in a recursive manner until the subset at a node all has the same value of the target variable, or when generalization can no longer adds meaning to the predictions.

III. CLASSIFICATION USING DECISION TREE INDUCTION

The stability and quality regarding the data mining of large datasets are composed of the following

1) Generalization by attribute-oriented induction, to compress the training data. This includes storage of the generalized student data to allow users to view the various data abstractions

2) Relevance analysis, to remove unnecessary data attributes, thereby further compacting the training data.

3) Multilevel mining, which combines the induction of decision trees with knowledge in concept hierarchies.

A. *Attribute-Oriented Induction(AOI)*

Generalization is the way of processing the raw data of any database may contain many attributes to be reduced by considering only the required attributes and continuing the process [16]. Attribute-Oriented induction is generalizing the data in to the required format that can replace primitive data into higher types (attribute removal). It allows the user to view the data at more meaningful abstractions. Furthermore, scalability issue is addressed in attribute-oriented induction by compressing the training data. In this paper we considered an Educational campus-placement center example, in which an attribute having all sub-nodes that has a class label "yes" can be replaced by a single node with class label "yes" and can check the other attributes against the conditions assumed. Attribute removal further compact the training data and reduces the bushiness of the resulting trees. The degree of generalization is controlled by a threshold. If the number of distinct values of an attribute is less than or equal to this threshold, then further generalization of the attribute is halted. Hence, attribute oriented induction not only provides increasing efficiency, also results in classification trees that are more understandable, smaller, and easier to interpret than trees obtained from methods operating on ungeneralized sets of low-level data.

B. *Relevance Analysis*

The uncertainty coefficient U(A) [17] for attribute A is used to further reduce the size of the generalized training data. U(A) is obtained by normalizing the information gain of A so that U(A) ranges from 0 (meaning statistical independence between A and the classifying attribute) to 1 (strongest degree of relevance between the two attributes). The user has the option of retaining either the n most relevant attributes or all attributes whose uncertainty coefficient value is greater than a pre-specified uncertainty threshold, where n and the threshold are user-defined. Note that it is much more efficient to apply the relevance analysis [13] to the generalized data rather than to the original training data

$$U(A) = \frac{I(P_{1,} P_2, \ldots, P_m) - E(A)}{I(P_1, P_2, \ldots, P_m)}$$

where $I(P_1, P_2, \ldots, P_m) = -\sum_{i=1}^{m} P_i P \log P_i / P$

and $E(A) = \sum_{j=1}^{k} \frac{P_{1j} + \ldots + P_{mj}}{P} I(P_{1j}, \ldots, P_{mj})$

Here P is the set of the final generalized training data, where P contains m distinct values defining with the output distinct output class Pi (for i = 1, 2, 3,…,m) and P contains P_i samples for each p_i, then the expected information needed to classify a given sample is I (P1, P2,…, Pm).

For example: we have the attribute A with the generalized final value {a_1 ,a_2 ,a_3 , ...,a_k} can be partition P into {C_1 ,C_2 ,C_3 , ... ,C_k} , where C_j contain those samples in C that have value a_j of A. The expected information based on partitioning by A is given by E(A) [17] equation, which is the average of the expected information. The gain (A) is the difference of the two calculations. If the uncertainty coefficient for attribute A is 0, it means no matter how we partition the attribute A, we won't get lose information. So the attributes A has no effect on the building of the final decision tree. If U (A) is 1, it means that we can use this attribute to classify the final decision tree. This is similar to find the max goodness in the class to find which attribute we can use to classify the final decision tree.

C. *Multilevel Mining*

Multilevel Mining method mines the data at different levels in the form of a decision tree which serves the desired outcome. It takes the input as Attribute oriented induction output and mines. The AVL-Tree will only support the decision making, while remaining all the other conditions have to be further generated. The information gain in attribute selection criterion has a tendency to favor multi valued attributes [17].The induction of decision trees is done at different levels of abstraction by employing the knowledge stored in the concept hierarchies. Furthermore, once a decision tree has been derived, the concept hierarchies can be used to generalize individual nodes in the tree and can reclassify data for the newly specified abstraction level. Generalization to very high concept levels can result in decision trees of little use since overgeneralization may cause the loss of interesting and important sub concepts.

IV. DECISION TREE CONSTRUCTION

The main idea of this paper is to construct a decision tree based on the proposed steps and prune it accordingly. The basic Decision Tree Construction Algorithm 1 is shown in Figure 1 which constructs a decision tree for the given training data.

Apart from *generalization threshold*, we also use two other thresholds for improving the efficiency namely, *exception threshold (€)* and *classification threshold (κ)*. Because of the recursive partitioning, some resulting data subsets may become so small that partitioning them further would have no statistically significant basis. These insignificant data subsets are statistically determined by the *exception threshold*. If the portion of samples in a given subset is less than the threshold, further partitioning of the subset is halted. Instead, a leaf node is created which stores the subset and class distribution of the subset samples.

In this process, the candidate with maximum information gain is selected as "test" attribute and is partitioned. The conditions, whether the frequency of the majority class in a given subset is greater than the classification threshold, or whether the percentage of training objects represented by the subset is less than the exception threshold, are used to terminate classification. Otherwise further classification will be performed recursively.

We consider a simple example to explain all the detail steps to generalize the final classification tree and find out the classification rules. Table I depicts a raw training data of education level in relation with the merit and skills.

Step 1 : Consider the student data as training data set

Stid	UG /PG	Previous history of study based on merit	age	Back logs if any	Company type	AOI	Category
100	UG	B.Tech	20	0	software	Higher studies	Yes
101	UG	B.Tech	20	1	software	job	No
102	UG	B.Tech	21	0	Software	job	Yes
103	UG	B.Tech	20	2	Core	Higher studies	yes
-	-	-	-	-	-	-	-

TABLE I. TRAINING DATA AS STUDENT DATA

Step2: Perform generalization by AOI method

Stid	Degree %	Age	Back logs	Company type	AOI	category
100	75	20	0	software	Higher studies	Yes
101	72	20	1	software	job	No
102	69	21	0	Software	job	Yes
103	60	20	2	core	Higher studies	yes
-	-	-	-	-	-	-

TABLE II. GENERALIZED TRAINING STUDENT DATA

The design procedure is implemented with the real world information about a study on educational campus and its placements criteria. In this regard, we started with a flowchart how the generalization of a large dataset can be done and it is depicted above in Table II.

Figure1 shows the proposed flow diagram to study and perform classification on large dataset.

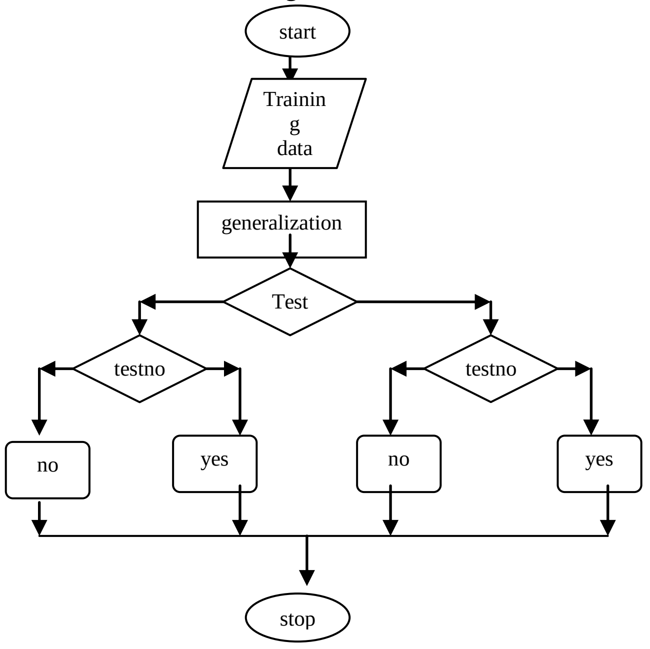

Figure 1. Proposed Flow chart representing classification on large Dataset

The training data is compressed with the storage of the generalized data which is done in two phases by attribute-oriented induction and multilevel induction.

Attribute-oriented induction is a knowledge discovery tool which allows the mining of large databases. Firstly, it allows the raw data to be handled at higher conceptual levels. Generalization of the training data is achieved by replacing primitive level data by higher level concepts. This induction method allows the user to view the data at more meaningful abstractions which optimizes the scalability issue by compressing the training data. After generalization, the training data will be much more compact than the original training data and still involves fewer input/output operations. Sometimes generalization involves removal of attributes without affecting the original database which further compresses the bushiness of the resulting trees.

To represent the data in user's view, the attribute-oriented induction results in classification trees that is more understandable, smaller, and easy for interpreting sets of low-level data. The degree of generalization is controlled by generalization threshold. Multilevel induction is the last step in generalization which combines the data from different levels of abstraction obtained by attribute-oriented induction with the knowledge stored in the hierarchies. Once a decision tree has been derived, the concept hierarchies can be used to generalize individual nodes in the tree and can reclassify data for the newly specified abstraction level.

The main idea of this paper is to construct a decision tree based on these proposed steps and prune it accordingly. The basic Decision Tree Construction Algorithm 1 is shown in Figure II, which constructs a decision tree for the given training data. Decision trees are data mining technology that has been around in a form very similar to the technology of today for almost twenty years now and early versions of the algorithms date back in the 1960s. Often times these techniques were originally developed for statisticians to automate the process of determining which fields in their database were actually useful or correlated with the particular problem that they were trying to understand.

Algorithm 1: Decision Tree Construction

```
DecisionTree (Node n, DataPartition D)
{
Apply AOI-Method to D to find
splitting-criterion of node n
Let k be the number of children of n
if k>O do
Create k children c1, c2,..., ck of n
Use splitting-criterion to partition D into D1,
D2..., Dk
for i = 1 to k do
DecisionTree(ci, Di)
end for
end if
Assign priority to the nodes based on the level;
}
```

Figure 2. Decision Tree

Step 3 : Data has to be mined at different levels for the users to view multiple abstractions of data.

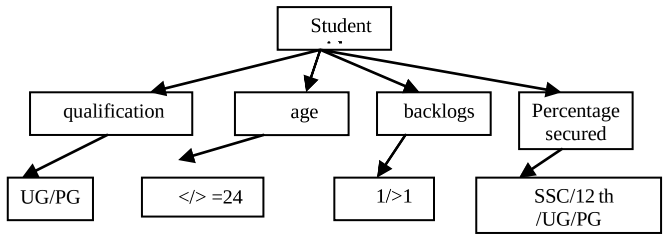

Figure 3. Decision Tree Construction for the Training Data

The Table III shows the selected data among the training data for placement of students in terms of quality and stability.

Stid	Company type	Class Label
102	Software	Yes
-	-	-

TABLE III SAMPLE OUTPUT OBTAINED FROM THE TRAING DATASET

V. DECISION TREE METHOD WITH AVL TREES

In this paper a decision tree is constructed based on the proposed steps and prune it accordingly and is shown in section IV, which constructs a decision tree for the given training data. Furthermore, once a decision tree has been derived with the proposed decision tree creation algorithm, the tree is formed as shown in Figure 3 with the concept hierarchies that can be used to generalize individual nodes in the tree and can reclassify data for the newly specified abstraction level.

Therefore, we proposed an algorithm called Node Merge, which allows merging of nodes in the tree thereby discouraging over- partitioning of the data. This algorithm also uses the concept of Height-Balancing in the tree using AVL trees depending on the priority checks for every node. This enhances the overall performance and final decision tree constructed is efficient enough to derive the classification rules effectively.

Algorithm 2 : Node Merge

```
Node_Merge( Node Data_A, Node Data_B)
{
Check priorities for node _A and node _ B;
if both the priorities > checkpoint then
{
Link _AB = remove _ link_ joining (Node Data _ A,
Node Data _B);
union = Node Data _ A. merge _with(Node Data _ B);
for (related node: nodes _ incident _to _either (Node Data _ A,
Node Data _B))
link _RA = link _ joining (related _node, Node Data _ A);
link _RB = link _joining (related _ node, Node Data _ B);
disjoin (related _ node, Node Data _ A);
disjoin (related _ node, Node Data _ B);
join (related _ node, union, merged _ link);
}
else print (Node have high priority, cannot be merged);
```

Figure 4. Decision tree construction

Algorithm 3 : To perform height balance

```
Perform _ balance _height (union, link _AB)
1.Check the tree obtained is in balanced.
2. if found then check the balance factor of the left/right sub
tree is heavy on left /right
3. if tree's right sub tree is heavy "left" then perform double
"left" rotation else
                    Perform single "left" rotation
4. if tree's left sub tree is heavy "right" then perform double
"right" rotation else
                    Perform single "right" rotation
5.Check for path preservations
```

Figure 5. Height-Balancing using AVL Tree Concept

This algorithm also uses the concept of Height-Balancing in the tree using AVL trees depending on the priority checks for every node. This enhances the overall performance, as the final decision tree constructed is efficient enough to derive the classification rules effectively.

Right-Right case and *Right-Left case*: when the balance factor of R is found as -2 , then the "right" sub tree is heavier than the left sub tree of the given node and it needs to be balanced by checking the balance factor, r of the right child. If r is less than zero then apply single left rotation with respect to R as root. If r is +1 then apply double left rotation (first rotation will be right as r as root and second rotation is a left rotation as R as root)

Left-Left case and *Left-Right case*: when the balance factor of R is found as +2 , then the "left" sub tree is heavier than the right sub tree of the given node and it needs to be balanced by checking the balance factor, r of the left child. If r is greater than zero then apply "single right" rotation with respect to R as root. If r is -1 then apply "double right" rotation (first rotation will be left as r as root and second rotation is a right rotation as R as root). This approach is shown in Figure 6.

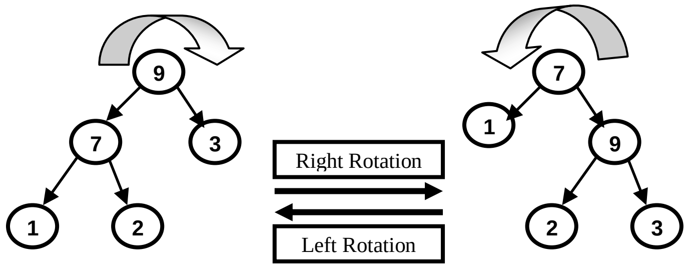

Figure 6. AVL Tree Right and Left single rotations

The final Decision Tree is constructed by using the above Algorithm 3, Balance Height. From the figure 7, it is clear that tree is well constructed and also balanced at every node.

As mentioned in the algorithm, the path to different levels are updated and preserved accordingly. In this way improved scalability and efficiency of the data classification with Decision Tree enhancement.

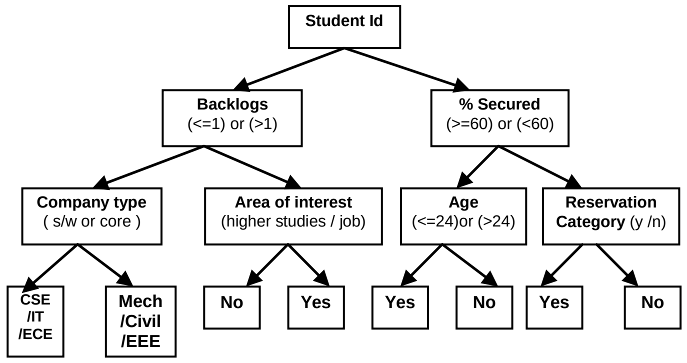
Student Id
Backlogs (<=1) or (>1)
% Secured (>=60) or (<60)
Company type (s/w or core)
Area of interest (higher studies / job)
Age (<=24)or (>24)
Reservation Category (y /n)
CSE /IT /ECE
Mech /Civil /EEE
No
Yes
Yes
No
Yes
No

Figure 7. Final decision tree with AVL tree concept

VI. Extensions and Conclusions

This paper proposes a approach for classification using Decision Tree Induction and it clearly shows how the algorithm generalizes the concept hierarchies from the training data by attribute-oriented induction (AOI). By generalization of the training data, it minimizes the requirement of the training data and makes the decision tree result meaningful using the AVL trees concept. The proposed algorithm provides a general framework that can be used with any existing Decision Tree Construction algorithms. In an effort to identify and rectify the restriction that limits the efficiency and scalability of other algorithms, we have proposed an efficient yet simple solution which will overcome them. Our future work involves further refinement in different applications of the proposed algorithm.

References

[1] W. J. Frawley, G. PiatetskyShapiro, and C. J. Matheus.Knowledge discovery in databases: An overview. In G. PiatetskyShapiro and W. J. Frawley, editors, *Knowledge Discoveryin Databases*, pages 1–27. AAAI/MIT Press,1991

[2] W. J. Frawley, G. PiatetskyShapiro,and C. J. Matheus.Knowledge discovery in databases: An overview. In G. PiatetskyShapiro and W. J. Frawley, editors, *Knowledge Discovery in Databases*, pages 1–27. AAAI/MIT Press,1991

[3] G. PiatetskyShapiro and W. J. Frawley. *Knowledge Discovery in Databases*. AAAI/MIT Press, 1991.

[4] L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification of Regression Trees*. Wadsworth, 1984.

[5] U. M. Fayyad, S. G. Djorgovski, and N. Weir. Automating the analysis and cataloging of sky surveys. In U. Fayyad, G. PiatetskyShapiro, P. J. yth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 471–493. AAAI/MIT Press, 1996.

[6] R. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.

[7] S. M. Weiss and C. A. Kulikowski. *Computer Systems that Learn: Classification and Prediction Methods from Statistics,Neural Nets, Machine Learning, and Expert Systems*.Morgan Kaufman, 1991.

[8] L. B. Holder. Intermediate decision trees. In *Proc. 14th Intl. Joint Conf. on Artificial Intelligence*, pages 1056–1062, Montreal, Canada, Aug 1995.

[9] M. Mehta, R. Agrawal, and J. Rissanen. SLIQ: A fast scalable classifier for data mining. In *Proc. 1996 Intl. Conf. on Extending Database Technology (EDBT’96)*, Avignon, France, March 1996.

[10] J. Shafer, R. Agrawal, and M. Mehta. SPRINT: a scalable parallel classifier for data mining. In *Proc. 22nd Intl. Conf. Very Large Data Bases (VLDB)*, pages 544–555, Mumbai (Bombay), India, 1996.

[11] J. Han, Y. Cai, and N. Cercone. Datadriven Discovery of quantitative rules in relational databases. *IEEE Trans.Knowledge and Data Engineering*, 5:29–40, 1993.

[12] J. Han and Y. Fu. Exploration of the power of attributeoriented induction in data mining. In U. Fayyad,G. PiatetskyShapiro,P. Smyth, and R. Uthurusamy, editors, *Advancesin Knowledge Discovery and Data Mining*, pages 399–421.AAAI/MIT Press, 1996.

[13] D. H. Freeman, Jr. *Applied Categorical Data Analysis*. Marcel Dekker, Inc., New York, NY, 1987.

[14] V. K. Vaishnavi, "Multidimensional height-balanced trees," IEEE Trans. Comput., vol. C-33, pp. 334-343, 1984

[15] Tomoki Watanuma, Tomonobu Ozaki, and Takenao Ohkawa. ―Decision Tree Construction from Multidimensional Structured Data. Sixth IEEE International Conference on Data Mining – Workshops, 2006.

[16] J. Han, Y. Cai, and N. Cercone. Datadriven discovery of quantitative rules in relational databases. IEEE Trans. Knowledge and Data Engineering, 5:29–40, 1993.

[17] Micheline Kamber, Lara Winstone, Wan Gong, Shang Cheng, Jiawei Han, ―Generalization and Decision Tree Induction: Efficient Classification in Data Mining‖, Canada V5A IS6, 1996.

[18] XindongWu · Vipin Kumar · J. Ross Quinlan · Joydeep Ghosh · Qiang Yang · Hiroshi Motoda · Geoffrey J. McLachlan · Angus Ng · Bing Liu · Philip S. Yu · Zhi-Hua Zhou · Michael Steinbach · David J. Hand · Dan Steinberg. A survey paper on ―Top 10 algorithms in data mining‖ 2007.