# A Technical Approach on Large Data Distributed Over a Network

Suhasini, G[1], Mamtha Billa[2], Ashwini, P[3]

[1]*Ganapathy College of Engineering, Warangal, India.*
[2]*Ramappa Engineering college, Warangal, India.*
[3]*Vaagdevi Engineering College,Warangal,India.*

e-mail: billa.mamatha@gmail.com

**Abstract - Data mining is nontrivial extraction of implicit, previously unknown and potential useful information from the data. For a database with number of records and for a set of classes such that each record belongs to one of the given classes, the problem of classification is to decide the class to which the given record belongs. The classification problem is also to generate a model for each class from given data set. We are going to make use of supervised classification in which we have training dataset of record, and for each record the class to which it belongs is known. There are many approaches to supervised classification. Decision tree is attractive in data mining environment as they represent rules. Rules can readily expressed in natural languages and they can be even mapped o database access languages. Now a days classification based on decision trees is one of the important problems in data mining which has applications in many areas. Now a days database system have become highly distributed, and we are using many paradigms. we consider the problem of inducing decision trees in a large distributed network of highly distributed databases. The classification based on decision tree can be done on the existence of distributed databases in healthcare and in bioinformatics, human computer interaction and by the view that these databases are soon to contain large amounts of data, characterized by its high dimensionality. Current decision tree algorithms would require high communication bandwidth, memory, and they are less efficient and scalability reduces when executed on such large volume of data. So there are some approaches being developed to improve the scalability and even approaches to analyse the data distributed over a network.**

**[keywords: Data mining, Decision tree, decision tree induction, distributed data, classification]**

## 1. Introduction

Decision tree is one of the most upcoming area of research as data goes on increasing we to select best attribute to classify the data[1].There are many data mining techniques that are available in order to draw the relationships between objects[2]. Decision tree is on of the widely used practical method [3].In this paper we present the overview of decision trees, decision tree induction algorithm, distributed data base, how the decision tree can be applied on distributed data.

## 2. Data Mining

Data mining is mining knowledge from large amount of data or knowledge discovery from data (KDD).
data mining task can be divided into two categories descriptive and predictive..

Descriptive data mining task characterize the general properties of data in data base.
Predictive data mining task performs inferences on current data in order to make predictions.
Data mining functionalities:
1. Data characterization: It summarizes the data of class under study.
2. Data discrimination: It is comparison of general features of target class with one or set of comparative classes
3. Association analysis: It is discovery of association rules showing attribute value condition that occur frequently together in a given set of data.
4. Classification: It is a process of finding a model. That describes and distinguishes data classes for the purpose of using the model to predict the class object whose class label is unknown. the derived model is based on training data .
The derived model can be represented in various forms
1. IF-THEN rules
2. Decision tree
3. Neural networks

1. IF-THEN rules:
Based on the condition of Association rules the attribute will be selected.
2. Decision tree:
It is a flow chart like tree structure, where each node denotes a test on a attribute value, each branch represents an outcome of the test and tree leaves represent classes or class distributions.
3. Neural networks: It is a collection of neurons like processing units with weighted connection between units.

Classification predicts categorical labels (discrete, unordered). Prediction models continuous data.

4. Clustering analysis: It analyses data objects without consulting a known class label.

5. Outlier analysis: The data base may contain data objects hat do not comply with general behaviour of data. These data objects are called as outlier. Most data mining methods discard outlier as noise or exception. The analysis of outlier data is outlier analysis.

6. Evolution analysis: This describes and models regularities and trends for object whose behavior changes over time[4].

**3. Decision trees**

A decision tree is a predictive model that, as its name implies, can be viewed as a tree. Specifically each branch of the tree is a classification question and the leaves of the tree are partitions of the dataset with their classification.

A decision tree is a data structure with the following properties

1. Each leaf is labeled with name of class

2. The root and each inter node are labeled with the name of attribute.

3. Every internal node has a set of two children are labeled with set of values of that nodes attribute such that union of all these constitute the set of all possible values for that attribute [5].

**3.1 Decision tree induction**

During 1970's and 1980's J.Ross Quinlan, a researcher in machine language developed a decision tree algorithm Known as ID3 (Iterative dichotomize).

Quinlan later presented C4.5.In 1984 a group of statistician's published a book on CART which describes the generation of binary tree

All the three algorithms adopt greedy approach in which decision tree are constructed in top down recursive divide and conquer manner [6].

**The basic decision tree algorithm steps**

Algorithm Generate_decsiontree (D, attribute list, attribute selection method)

1. Create a Node N;
2. if tuples in D are all of same class, C then
3. Return N as a leaf node labelled with the class C;
4. if attribute list is empty then
5 return N as a leaf node labelled with the majority class in D;
6 apply attribute selection methods to find best splitting criteria
7 label node N with splitting criteria;
8 if splitting attribute is discrete and multiway split allowed then
9 attribute list<- attribute list-splitting attribute;
10 for each outcome of j of splitting criteria
11 let $D_j$ be the set of data tuples in D satisfying outcome j;
12 if $D_j$ is empty then
13 attach a leaf labelled with the majority class in D to node N;
14 else attach the node returned by Generate_decsiontree (D, attributelist) to node N;
End for
15 return N

**3.2 Attribute selection measures**

ID3 uses Information gain as its attribute selection measure.

Let D, the data partition.supose the class label attribute has M distinct values defining m distinct clases,$C_i$ (for i=1...m)
Let $C_{i,D}$ denote set of tuples in class $C_i$ in D
Let |D| and | $C_{i,D}$| denote the number of tuples in D and $C_{i,D}$ respectively.
the information needed to classify a tuple in D is given by

$$Info\ (D)=-\Sigma^{m}\ i_{=1}p_i\log_2(p_i)$$

The attribute with highest gain is chosen as the splitting attribute for node N.

$p_i$ is the probability that an arbitrary tuple in D belongs to class $C_i$ and is estimated by

$$|\ C_{i,D}\ |/|\ D\ |$$

now we need to partition the tuples in D on some attribute A having v distinct values,{a1,a2,….av}.If A is discrete value then attribute A can be used o split D into V partitions{D1,D2…Dv} where $D_j$ contains those tuples in D that has the outcome $a_j$ of A.

Even though the will be impure in order to arrive an exact classification the needed information is got by

$$Info_A(D)=\Sigma_{j=1}^{v}|\ D_j\ |/|\ D\ |\ *\ info(D_j)$$

Information gain is defined as difference between original information requirement and the new requirement

$$Gain\ (A)=info(D)-info_A(D)$$

By using information gain on some tuple results in large partitions since the partition is pure info(D)=0 since information gained would be max so it is not apt for classification

C4,5 make use of gain ratio,it applies a normalization to information gain by using split information

$$splitinfo_A(D)=-\ \Sigma_{j=1}^{v}|\ D_j\ |/|\ D\ |*\log_2(|\ D_j\ |/|\ D|)$$

Gain ratio=Gain (A)/Split info (A)
Splitinfo may also reach 0, which is unstable

CART makes use of Gin index as splitting criteria.

$$Gini\ (D)=1-\Sigma^{m}{}_{i=1}p_i^{2}$$

When considering a binary split we compute the weighed sum of impurity of each resulting partition

for example if a binary split on a partition D into D1, D2 then Gininindex is given by
$Gini_A(D)=|D_1|/|D| Gini(D_1)|D_2/|D| Gini(D_2)$

We need to select a measure that tends to produce shallower tree.

### 3.3 Tree Pruning

Many of the branches will reflect anomalies in the training data due to noise or outliers. They are typically used to remove less reliable branches. Pruned tree is small and less complex and thus easier to comprehend. There are two common approaches: 1 pre pruning 2 postprunind. Prepruning: a tree is pruned by halting its construction early. Post pruning: It removes subtrees from fully grown tree.

### 3.4 Scalability and decision tree induction:

The existing decision tree algorithms such as ID3, C4.5, and CART are well established to handle relatively small data.
here is a restriction that the training tuples should be within memory. Since there are large training tuples available, the decision tree construction becomes inefficient.
SLIQ and SPRINT are algorithms for induction of decision tree on large data sets.
Both make use of a different data structure.
SLIQ employs disk resident attribute list and memory resident class list.
class list remains in memory, but when class list increases and do not fit in the memory then performance of SLIQ decreases.
SPRINT makes use of different data structure that holds class and attribute information.
SPRINT removes all memory restriction yet requires hash trees. They may become expensive as training set size grows.
To enhance the scalability a method called Rain Forest was proposed it can be applied on any decision tree. It makes use of AVC-set (Attribute –value, class label) for each attribute at each node.
BOAT (Bootstrapped Optimistic Approach for Tree construction) It does not make use of any data structure but makes use of statistical technique o to create several smaller samples of training data which will fit in memory.

### 3.5 Applications of decision trees

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. Partially because of this history, decision tree algorithms tend to automate the entire process of hypothesis generation and then validation much more completely and in a much more integrated way than any other data mining techniques. They are also particularly adept at handling raw data with little or no pre-processing. Perhaps also because they were originally developed to mimic the way an analyst interactively performs data mining they provide a simple to understand predictive model based on rules (such as “90% of the time credit card customers of less than 3 months who max out their credit limit are going to default on their credit card loan.”).

Because decision trees score so highly on so many of the critical features of data mining they can be used in a wide variety of business problems for both exploration and for prediction. They have been used for problems ranging from credit card attrition prediction to time series prediction of the exchange rate of different international currencies. There are also some problems where decision trees will not do as well. Some very simple problems where the prediction is just a simple multiple of the predictor can be solved much more quickly and easily by linear regression. Usually the models to be built and the interactions to be detected are much more complex in real world problems and this is where decision trees excel [7].

## 4. Distributed Database

A **distributed database** is a database in which storage devices are not all attached to a common cpu. It may be stored in multiple computer located in the same physical location, or may be dispersed over a network of interconnected computers.

Collections of data (e.g. in a database) can be distributed across multiple physical locations. A distributed database can reside on network servers on the internet, on corporate internet or on other company networks. The replication and distribution of databases improves database performance at end users worksites.

To ensure that the distributive databases are up to date and current, there are two processes: replication and duplication. Replication involves using specialized software that looks for changes in the distributive database. Once the changes have been identified, the replication process makes all the databases look the same. The replication process can be very complex and time consuming depending on the size and number of the distributive databases. This process can also require a lot of time and computer resources. Duplication on the other hand is not as complicated. It basically identifies one database as a master and then duplicates that database. The duplication process is normally done at a set time after hours. This is to ensure that each distributed location has the same data. In the duplication process, changes to the master database only are allowed. This is to ensure that local data will not be overwritten. Both of the processes can keep the data current in all distributive locations.

Besides distributed database replication and fragmentation, there are many other distributed database design technologies. For example, local autonomy, synchronous and asynchronous distributed database technologies. These technologies’ depend on the needs of the business and the sensitivity/confidentiality of the data to be stored in the database, and hence the price the

business is willing to spend on ensuring data security and consistency[8] .

Distributed computing plays an important role in the Data Mining process for several reasons. First, Data Mining often requires huge amounts of resources in storage space and computation time. To make systems scalable, it is important to develop mechanisms that distribute the work load among several sites in a flexible way. Second, data is often inherently distributed into several databases, making a centralized processing of this data very inefficient and prone to security risks. Distributed Data Mining explores techniques of how to apply Data Mining in a non-centralized way[9].

One interesting aspect of large databases is that they are often distributed over many locations. The main reason for this is that they are produced by a variety of independent institutions. While these institutions often allow a second party to browse their databases, they will rarely allow this party to copy them. There could be a number of reasons for this: the need to retain the privacy of personal data recorded in the database, through questions regarding its ownership, or even because the sheer size of the data makes copying non permissively costly in CPU, disk I/O or network bandwidth [10].

We require distributed algorithms for data mining over a distributed network as the data is distributed on various locations.A distributed decision tree induction algorithm is one that executes on several computers, each with its own database partition. The outcome of the distributed algorithm is a decision tree which is the same as, or at least comparable with, a tree that would be induced were the different partitions collected to a central place and processed using a sequential decision tree induction algorithm.

DHDT (Distributed hierarchical decision tree) focuses on reducing the volume of data sent from each level to the next while preserving perfect accuracy.

The common approach to reduce the communication overhead would be to sample the distributed data set and transfer these decision trees to the central sites this approach is called as Meta learning. This suffers from scalability limitations. So a different Meta learning algorithm was suggested this algorithm turns each decision tree to set of rules and then merge the rules into single superset of rules. As numbers of sites increase the accuracy of Meta learning classifiers drop.

Then much attention was given to parallel algorithm There are three parallel algorithms

1. Synchronous tree construction
2. Partitioned tree algorithm
3. Hybrid approach

These algorithms cannot be used in large scale distributed system because data movement is often impractical in distributed network

The parallel version of SPRINT algorithm enhances the performance by vertical partitioning scheme where every computing node is responsible for a distinct subset of data attribute. In order to split the attribute list the hash table must be available on all computing nodes which make the algorithm highly unscalable[11].

**4.1 Bounds on Gain Function:**

By these bounds we can avoid collecting the cross table of many of the attributes whose gain as indicated by the bounds cannot be large enough to change the result.

**4.2 Notations**:

Let P be the population of size n

And let {P1, P2} be the partition of P into two sub population of size n1,n2.

Let the cross table of the P1, P2, P are as follows

$$\overrightarrow{P1(\text{values, class})} = \begin{pmatrix} a11 & a12 \\ a21 & a22 \end{pmatrix}$$

$$P2(\text{values, class}) = \begin{pmatrix} b11 & b12 \\ b21 & b22 \end{pmatrix}$$

$$P(\text{values, class}) = \begin{pmatrix} a11+b11 & a12+b12 \\ a21+b21 & a22+b22 \end{pmatrix}$$

Here aij, bij denote the number of learning examples with value i, j

**4.3 Gini index function:**

Given giniindex (P1), giniindex (P2), n1, n2

Then upperbound on giniindex (P) is given by

upperbound=n1*giniindex(P1)+n2*giniindex(P2)/n1+n2

P1, P2, n1, n2 are given then cardinality decision split P1 in to two subsets $P1^{left}$ $P2^{right}$ with sizes $n1^{left}$ and $n2^{right}$ then lower bound is given by

Lowerbound=

Gain index(P1)/[1+n2/n1][1+max{$n/n1^{left}$,$n2/n1^{right}$}]

The upper bound on information gain is given by

n1*infogain(P1)+n2*infogain(P2)/n1+n2

The lower bound on information gain is given by

$$\left[\frac{1}{1+(n2/\min(n1,n1))}+\frac{1}{1+n2/n1}\right]$$

*infogain(P1)

let P be the population of size n and {P1,P2,P3…PK} a partition of P into k sub populations of size n1,n2,n3….nk .let G() be the gain function then upper bound on G(P) is given by

$$G(P) \leq \frac{\Sigma^{k}\, i_{=1}\ n_i G(P_i)}{\Sigma^{k}_{\ i=1}\ n_i}$$

let P be the population of size n,and {P1,P2} a partition of p into two subpopulation of size n1,n2 and the candidate split divides P1 into two subset $P1^{left}$ ,$P1^{right}$ with size $n1^{left}$ ,$n1^{right}$ let G() denote the gain function the lower bound is given by

$$G(P) \geq \frac{G(P1)}{[1+n2/n2][1+n2/\min(n1lt,n1ri)]}$$

**5. Distributed HierarchicalDecision tree.**

The distributed hierarchical decision tree (DHDT) algorithm runs on a group of computers, connected through a wide-area network such as the Internet. Each computer has its own local database. The goal of DHDT is to derive exactly the same decision tree learned by a sequential decision tree learner on the collection of all data in the network. We assume a homogeneous database schema for all databases, which can be provided transparently, if required, by ordinary system services. The algorithm relies on a (possibly overlay) communication tree that spans all computers in the group. The communication tree can be maintained by a spanning tree algorithm or can utilize then actual hierarchy of the network. For reasons of locality, communication between nodes in the lower levels of the spanning tree is often cheaper than communication between nodes in the upper levels. Thus, a good algorithm will use more communication at the bottom than at the top of the tree. We further assume that during the growth phase of the decision tree, the databases and the communication tree remain static.

For each data base, an entity called agent is allocated, which accesses the database through a standard interface such as SQL or HL7 and gathers simple statistics. The agent is in charge of computing the required statistics from local database and participating in distributed algorithm.

The root agent is responsible for developing the decision tree and making the split decisions for the new decision tree leaves. First, the root agent decides whether a decision tree leaf has to be split according to one or more stopping conditions (e.g., if the dominance of the majority class has already reached a certain threshold) or according to the PUBLIC method ,which avoids splitting a leaf once it knows it may which holds the number of examples that belong to

each distinct class in the population, is sufficient for computing these functions, and thus it is aggregated by the agents over the communication tree to the root agent.

Recall that if a decision tree leaf has to be split, thus a split must be done by the attribute with the highest gain in the combined database of the entire network. All that is required to decide on the splitting attribute is an agreement as to which attribute has the maximal gain; the actual gain of each attribute does not need to be computed. To reach agreement, the agents participate in a distributed algorithm called DESAR (Distributed Efficient Splitting Attribute Resolver). For each new leaf that has to be developed, DHDT starts a new instance of DESAR to and the best splitting attribute [12]

**5.1 DHDT ALGORITHM**:

Algorithm: the DHDT algorithm for root agent
Initialization
New leaves list=decision tree root
Algorithm

1. for each $leaf_i$ in new leaves list do
2. remove $leaf_i$ from new leaves list
3. $attribute_k$ =run DESAR for $leaf_i$
4. if the gain form splitting $leaf_i$ according to $attribute_k$ is above the pruning threshold
5. split $leaf_i$ by $attribute_k$
6. insert a new leaves to new leaves list
7. end if
8. end

**5.2 Distributed Efficient Splitting Attribute Resolver** :

To no the best splitting attribute while minimizing communication complexity, DESAR aggregates only a subset of the attribute cross tables over the communication tree to the root agent. The algorithm starts when the agents receive a message that is broad-cast down the communication tree (initiated by the root and transmitted by each agent to all its children), asking for the development of a new leaf in the decision tree. Then, each agent waits for messages from its children. When messages are received from all children, the agent combines the received cross tables with its own lo-cal cross tables, picks the most promising attributes on the basis of its aggregated data, and sends the corresponding cross tables to its parent agent only the cross tables for these attributes.

Since different subtrees may choose to send information on different subsets of attributes to their root, the information eventually collected by the root does not always suffice to decide which attribute maximizes the gain function. Instead of assuming a single value we consider the lower bound and upper bound and compute attribute interval for each attribute. The interval between lower bound and upper bound for the gain of an attribute is called as attribute interval.

The bounds are computed using the information received from all agent's children. The bound computed by the root agent bound the gain function over all data in the network.

Using the attribute interval, we can say that the given threshold defines a clear separation of interval if it separates the attribute interval into two nonempty disjoint sets of interval and neither of interval crosses the threshold.

When bounds are computed the agent sets a border with a minimal number of attributes having their lower bound larger than border. The attribute whose interval lay above the border are called promising and their cross tables are sent to agents parent. If not the agent request more information for its children by naming the attribute for which more information is needed (naming method).

**DESAR ALGORIHM**:

Algorithm Definition

D1. border=maximal lower bound of all attribute which were not sent to parent

D2.border attribute= the attribute whose lower bound defines the border.

D3.if agent is root then

D4. extra condition=there is only a single attribute $A_i$

Where uperbound ($A_i$)>= border or

Max (upperbound ($A_i$)) =border

D5. else

D6. extra condition=$G^i_u$<border for all children.

Algorithm

On initialization new leaf is born

1. receive information from children
2. while(not(border defines clear separation and extra condition)) do
3. if($G^i_u$ >border) then
4. request child to lower its border and send new information
5. else if(border does not define a clear separation and cross table of border attribute has partial information)
6. request information for border attribute from children who did not sent complete information
7. else
8. request information for all attributes that cross the border
9. endif
10. receive information from all children
11. end while
12. Return attribute $A_i$ where lowerbound ($A_i$)>=border.

*On request for more information from parent*

1. if(parent requires more information for attribute $attr_i$) then
2. if(crosstable of $attr_i$ was not sent to parent) then
3. send parent the cross table of $attr_i$
4. else
5. request information for $attr_i$ from children who sent partial information regarding $attr_i$
6. else(the case where parent request that the border be lowered)
7. update border and border attribute and start phase 1
8. endif

**Computing the lower and upper bounds on gain of attribute**

Let agent1, agent2, agent3….. be the descendants of agent0

Let $child_i$ denote the ith immediate child of agent0

Let $P_d$ be the population of $agent_d$ derived from its local database

then the gain function over a network is defined as

$$P=U_d P_d$$

for any given attribute let

$P_u=U_{d=k+1----q}P_d$ be the combined population of descendant agents that did not send the attributes crosstable to agent0

Let $P_k=P/P_u$ where $P_k$ be the combined population of descendants who sent the attribute crosstables and are also the descendants of $child_i$ including the $child_i$

Let $P^i u$ be the combined population of descendant agents who did not send the cross tables but are descendants of child

G() denotes the gain function

**Upperbound**

T he agent computes an upper bound $G_u$ on G ($P_u$**)**

This is computed recursively by the child and sends to its parent the upper bound $G^i u$

It is computed by the following rule

1. if the attribute cross table is not sent to its parent $G^i u$ is equal to attributes upper bound ,otherwise $G^i u$ is equal to $G_u$ of the child itself
2. for the leaf agent $P_u$=□ thus $G_u$ is set to 0

Now by using the upperbound of giniindex function $G_u$

$G_u \geq G(P_u)$

Where

$$G_u = \frac{\Sigma_i |P^i u| G^i u}{\Sigma_i |P^i u|}$$

again applying the same recursively the agent computes the upper bounds as

$$G(P) \leq \frac{|P_k|G}{(P_k)+|P_u|G}$$

$|P|$ can be easily computed for aggregate class distribution vector and $|P_k|$, then $|P_u|$ can be computed

In order to further reduce the communication complexity and make it independent of number of candidate attribute a child agent sends the max $G^i_u$ of all attributes as single upper bounds denoted as $\mathbf{G^i_u}$

**Lower bound**

it is computed from lower bound of gini index function where $P1= P_k$ and $P2= P_u$

***Efficient request methods:***

There are two methods that may be used to efficiently request for more information

1 naming method

2. Independent method

If $\mathbf{G^i_u}$ of child is above the border the independent method is used. If clear separation does not exist and highest attribute has partial information the child uses naming method to request information for the highest attribute only, from all the children who sent partial information regarding the attribute. Reducing the message complexity. Since the request for more information is wasteful it is beneficial to send small number of additional attributes.

This DESAR algorithm is being enhanced by further lowering the border by a small constant $\epsilon$ .

***Cross Tables:***

Cross table (crosstab for short) is a two-way table consisting of rows and columns. It is typically used to determine whether there is a relation between row variables and column variables. Cross tables are often used in various names. For example, pivote tables and multidimensional tables are often used in Business intelligence (BI) and OLAP contexts. In essence, they are the same tables operating on the same principles. Normally they have a grand total, row totals, column totals, cell values, and sub-totals.

**Cross table of Agent A:**

| Attribute1 | | | Attribute 2 | | |
|---|---|---|---|---|---|
| | C1 | C2 | | C1 | c2 |
| V1 | 20 | 65 | V1 | 50 | 22 |
| V2 | 48 | 27 | V2 | 18 | 70 |
| Giniindex .5928 | | | Giniindex .63 | | |

**Cross table for agent B**

| Attribute1 | | | Attibute2 | | |
|---|---|---|---|---|---|
| | C1 | C2 | | C1 | c2 |
| V1 | 5 | 60 | V1 | 40 | 20 |
| V2 | 45 | 20 | V2 | 10 | 60 |
| Giniindex 0.716 | | | Giniindex 0.663 | | |

**Agent C combined information:**

| Attribute1 | | | Attribute2 | | |
|---|---|---|---|---|---|
| | C1 | C2 | | C1 | c2 |
| V1 | 20 | 65 | V1 | 0 | 0 |
| V2 | 48 | 27 | V2 | 0 | 0 |
| Giniindex 0.5928 | | | Giniindex 0.5 | | |

***Agent C complete information***

Agent A and Agent B are children of Agent C.After computing the gain of each attribute agent A sends only the cross table of its best splitting attribute,ie attribute 1,similarly agent B also sends the cross table of attribute 1.agent c combines the two and choose attribute 1 as best splitting attribute but correct thing is we need to pick attribute 2 which has highest gain when combined.So to get addition data on an attribute we need DESAR algoritm to request additional information so that we select the best splitting attribute.

**6. Conclusion**

In this paper we presented the classification of large data that is distributed over a network.and algoritms on how to select the splitting attribute. We need to even have efficient and scalable algorithms to classify data over a distributed data base.

**References**

[1]. Juanlihu, JiabinDeng,MinxiangSui,A new approach to decision treeBased on Principle componenet Analysis ,proceeding conferences of computational intelligence and software Engineering page no 1-4 2009.

[2].HuiminZhaoandathishPsinha, An efficient AlgorithmforgeneratingGeneraliseddecisionforests, IEEE transactionon Systems,Man,andcybernetics,Part-ASystemsand Humans VOL-35, pageno287-299 ,sep 2005.

[3] .D.Lui, C.Lai and W.Lee,AHybrid Of Sequential rules and collaborative filtering for product recomdation,Information sciences179(20) page no 3505-3519,2009.

[4]. J.c.shaer R.agarwaland M.mehtha SPRINT:"scalable parallel classifier for data minig"proc.22nd Intlconf.very large databases for data mining

[5]. M.Vjoshi, g.karypis, and v.kumar,"A new scalable and effient parallel classifier Algorithm for mining large data sets,"proc.int'l parallel processing symppp, 1998.

[6]. K alsabti, s.ranka and v.singh,"clouds: A decision tree classifier for large datasets", Knowledge discovery and data mining, pp2, 1998.

[7].A.srivasthava, E_H.s.Han, V.Kumar and Vsingh,"parallel formulation of decision tree classification algorithm", data mining and knowledge discovery .

[8] P.K Chan and S.j stolfo,"towards parallel and distributed learning by Meta learning,"Working notes AAAI works,Knowledge discovery in data bases.

[9] H.Karguptha, B Park, D hershbereger and E.johnson"collective data mining: A new perspective towards distributed data mining.

[10]. jaiweiHan and Micheline Kamber:"Data mining concepts and techniques".

[11].Amir Bar-or,Daniel keren Assaf schuter and Ran wolff"Hierarchical decision tree induction in distributed genomic data base":IEEE transaction on knowledge and data engineering.

[12] .Amir Bar-or, Daniel keren Assaf schuter"Hierarchical decision tree induction in distributed genomic data base": IEEE transaction on knowledge and data engineering.