Krishna Mohan, Surekha Alokam, MHM Krishna Prasad / International Journal of Engineering Research and Applications (IJERA) ISSN: 2248-9622 www.ijera.com
Vol. 2, Issue 3, May-Jun 2012, pp.1401-1405

An Efficient Decision Tree For Uncertain Data

Krishna Mohan[*1], Surekha Alokam [#1], MHM Krishna Prasad [#2]
[1,2] Associate professor, Dept. of Computer Science Jawaharlal Nehru Technological University
[*1]Kakinada, [#2]Vijayanagaram, Andhra Pradesh.
[#1] M.Tech, Computer Science Department, JNTU Kakinada, Andhra Pradesh.

Abstract: To handle known values and data we have traditional decision trees. Decision tree is used for classification technique. We propose a decision tree based classification for uncertain data. Data uncertainty is commonly exists in many applications during data collection, such as measurement/quantisation errors, data staleness, and multiple repeated measurements. With the use of uncertainty value of data is not represented by single value, but by multiple values forming a probability distribution. In this paper we extend traditional decision tree algorithms to handle both certain data and uncertain datasets. Here the uncertainty is handled by considering the probability density function (pdf). The resulting classifiers are more accurate than the traditional one. we can use these algorithms for handling both categorical data as well as numerical data also.

Keywords: Uncertain Data, Decision tree, Classification, Data Mining

I. Introduction.

Decision trees are a simple yet widely used method for classification and predictive modeling. A decision tree partitions data into smaller segments called terminal nodes. Each terminal node is assigned a class label[2,6,10]. The non-terminal nodes, which include the root and other internal nodes, contain attribute test conditions to differentiate each record from others that have different characteristics. This process terminates when the subsets cannot be partitioned further. In this paper we study how to control data uncertainty by using decision tree classification. A simple way to handle data uncertainty is to abstract probability distribution by means and variances. This approach is called Averaging. Here we have another approach also that is called Distribution based approach in this complete information is utilized. In this paper the Decision tree can handle both numerical and categorical data, while many other techniques are usually specialized in analyzing datasets that have only one type data. Our goals are 1) To invent an algorithm for building decision trees for uncertain data using Distribution based approach.
2) To check whether the Distribution approach could lead to higher accuracy when compared with averaging.
3) Pruning techniques are derived to significantly improve the computational efficiency of Distribution based algorithm.
Data uncertainty arises basically in many applications due to various reasons. We have three categories here: measurement errors, data staleness, and repeated measurements.

1) Measurement Errors: Data obtained from measurements by physical devices are often not precise due to measurement errors. For example, thermometer measures body temperature by measuring the temperature of the ear drum via an infrared sensor. It is having some calibration error.
2) Data Staleness: In some of the applications, data values are no longer fresh that means the values are continuously changing. Example for this is location based tracking system.
3) Repeated Measurements: Basically the most common uncertainty comes from repeated measurements. Example for this is a patient's body temperature could be taken multiple times during a day.

II. Handling Uncertainty

Under our uncertainty model, a feature value is represented not by a single value, vi,j , but by a pdf, fi,j. A decision tree under our uncertainty model resembles that of the point-data model. The difference lies in the way the tree is employed to classify unseen test tuples. Similar to the training tuples, a test tuple t0 contains uncertain attributes. Its feature vector is thus a vector of pdf's (f0,1,,,,,,,,f0,k).A classification model is thus a function M that maps such a feature vector to a probability distribution P over C. The probabilities for P are calculated as follows. During these calculations, we associate each intermediate tuple tx with a weight wx ϵ [0,1]. Further, we recursively define the quantity $ø_n(c; tx,wx)$, which can be interpreted as the conditional probability that tx has class label c, when the sub tree rooted at n is used as an uncertain decision tree to classify tuple tx with weight wx.

III. Algorithms

In this section, we discuss two approaches for handling uncertain data. The first approach, called "Averaging", transforms an uncertain dataset to a point-valued one by replacing each pdf with its mean value. A decision tree can then be built by applying a traditional tree construction algorithm. To exploit the full information carried by the pdf's, our second approach, called "Distribution-based", considers all the sample points that constitute each pdf.

A. Relative Averaging

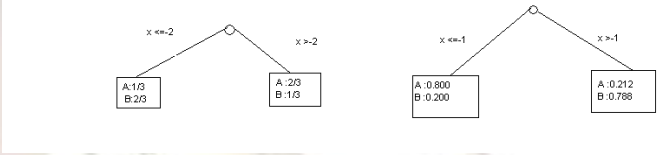
We simply replace each pdf with its expected value, thus effectively converting the data tuples to point-valued tuples. This reduces the problem back to that for point valued data, and hence C4.5 [3] (with pre-pruning and post pruning) can be reused. We call this approach AVG (for "averaging").

AVG is a greedy algorithm that builds a tree top-down. When processing a node, we examine a set of tuples S. The algorithm starts with the root node and with S being the set of all training tuples. At each node n, we first check if all the tuples in S have the same class label c. If so, we make n a leaf node and set $P_n(c) = 1$, $P_n(c') = 0$ for all $c' \neq c$. Otherwise, we select an attribute A_{jn} and a split point z_n and divide the tuples into two subsets: "left" and "right". All tuples with $v_{i,jn} \leq z_n$ are put in the "left" subset L; the rest go to the "right" subset R. If either L or R is empty (even after exhausting all possible choices of A_{jn} and z_n), it is impossible to use the available attributes to further discern the tuples in S. In that case, we make n a leaf node. Moreover, the population of the tuples in S for each class label induces the probability distribution P_n. In particular, for each class label c ϵC, we assign to $P_n(c)$ the fraction of tuples in S that are labelled c. If neither L nor R is empty, we make n an internal node and create child nodes for it. We recursively invoke the algorithm on the "left" child and the "right" child, passing to them the sets L and R, respectively.

To build a good decision tree, the choice of A_{jn} and z_n is crucial. At this point, we may assume that this selection is performed by a black box algorithm Best Split, which takes a set of tuples as parameter, and returns the best choice of attribute and split point for those tuples. We will examine this black box in details. Typically, Best Split is designed to select the attribute and split point that minimizes the degree of dispersion. The degree of dispersion can be measured in many ways, such as entropy (from information theory) or Gini index [4]. The choice of dispersion function affects the structure of the resulting decision tree. In this paper we assume that entropy is used as the measure since it is predominantly used for building decision trees. The minimizations is taken over the set of all possible attributes A_j (j = 1,…….,k), considering all possible split points in dom(A_j). Given a set S = {t_1,……., t_m} of m tuples with point values, there are only m-1 ways to partition S into two non-empty L and R sets. For each attribute A_j , the split points to consider are given by the set of values of the tuples under attribute A_j , i.e., {$v_{1,j}$,….,$v_{m,j}$}. Among these values, all but the largest one give valid split points.

Tuple	Class	mean	Probability	Distribution			
			-10	-1.0	0.0	+1.0	+10
1	A	+2.0		8/11			3/11
2	A	-2.0	1/9	8/9			
3	A	+2.0		5/8		1/8	2/8
4	B	-2.0	5/19	1/19		13/19	
5	B	+2.0			1/35	30/35	4/35
6	B	-2.0	3/11			8/11	

TABLE 1



(a) Relative Averaging (b) Distribution-based

Fig.1. Decision tree built from Example

B. Distubution-based Approach

For finding uncertain data, we adopt the same decision tree building framework as described above for handling point data. After an attribute A_{jn} and a split point z_n has been chosen for a node n, we have to split the set of tuples S into two subsets L and R. The major difference from the point-data case lies in the way the set S is split. Recall that the pdf of a tuple $t_i \epsilon$ S under attribute A_{jn} spans the interval [$a_{i,jn}$, $b_{i,jn}$]. If $b_{i,jn} \leq z_n$, the pdf of t_i lies completely on the left of the split point and thus t_i is assigned to L. Similarly, we assign ti to R if $z_n < a_{i,jn}$. If the pdf properly contains the split point, i.e., $A_{i,jn} \leq z_n < b_{i,jn}$, we split ti into two fractional tuples t_L and t_R in the same way as described in previous Section and add them to L and R, respectively.

The key to building a good decision tree is a good choice of an attribute A_{jn} and a split point z_n for each node n. With uncertain data, however, the number of choices of a split point given an attribute is not limited to m-1 point values. This is because a tuple t_i's pdf spans a continuous range [$a_{i,j}$, $b_{i,j}$]. Moving the split point from $a_{i,j}$ to $b_{i,j}$ continuously changes the probability $p_L = \int_{a_{i,jn}}^{z_n} F_{i,jn}(x)\, dx$ (and likewise for p_R). This changes the fractional tuples t_L and t_R, and thus changes the resulting tree. If we model a pdf [9,13] by s sample values, we are approximating the pdf by a discrete distribution of s points. In this case, as the split point moves from one end-point $a_{i,j}$ to another end-point $b_{i,j}$ of the interval, the probability p_L changes in s steps. With m tuples, there are in total ms sample points. So, there are at most ms-1 possible split points to consider. Considering all k attributes, to determine the best (attribute, split-point) pair thus require us to examine k(ms -1) combinations of attributes and split points. Comparing to AVG, UDT is s time more expensive. Note that splitting a

tuple into two fractional tuples involves a calculation of the probability p_L, which requires an integration. We remark that by storing the pdf in the form of a cumulative distribution, the integration can be done by simply subtracting two cumulative probabilities. Let us re-examine the example tuples in Table I to see how the distribution-based algorithm can improve classification accuracy. By taking into account the probability distribution, UDT builds the tree shown in Figure 3 before pre-pruning and post-pruning are applied. This tree is much more elaborate than the tree shown in Figure 1(a), because we are using more information and hence there are more choices of split points. The tree in Figure 3 turns out to have a 100% classification accuracy! After post-pruning, we get the tree in Figure 1(b). Now, let us use the 6 tuples in Table I as testing tuples4 to test the tree in Figure 1(b). For instance, the classification result of tuple 3 gives P(A) = 5/8*0.80 + 3/8 * 0.212 = 0.5795 and P(B) = 5/8 * 0.20 + 3/8 * 0.788 = 0.4205. Since the probability for "A" is higher, we conclude that tuple 3 belongs to class "A". All the other tuples are handled similarly, using the label of the highest probability as the final classification result. It turns out that all 6 tuples are classified correctly. This hand-crafted example thus illustrates that by considering probability distributions rather than just expected values, we can potentially build a more accurate decision tree.

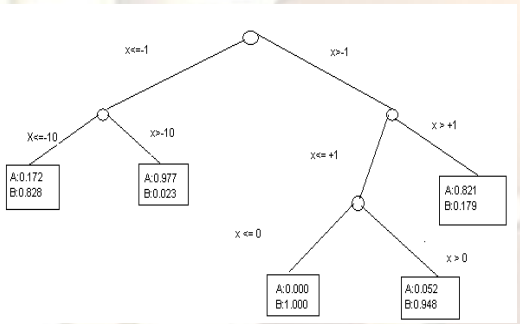


Fig.2. Example tree before post pruning

m=number of tuples, and s = number of samples per pdf. For each such candidate attribute A_j and split point z, an entropy $H(z,A_j)$ has to be computed . Entropy calculations are the most computation-intensive part of UDT. Our approach to developing more efficient algorithms is to come up with strategies for pruning candidate split points and entropy calculations. Note that we are considering safe pruning here. We are only pruning away candidate split points that give sub-optimal entropy values. So, even after pruning, we are still finding optimal split points. Therefore, the pruning algorithms do not affect the resulting decision tree, which we have verified in our experiments. It only eliminates sub-optimal candidates from consideration, thereby speeding up the tree building process.

C. Algorithm for Both Certain And Uncertain Data

Input: The training dataset D; the set of candidate attributes att-list

Output: An uncertain decision tree

Begin

1: create a node N;
2: if (D are all of the same class, C) then
3: return N as a leaf node labeled with the class C;
4: else if (attribute-list is empty) then
5: return N as a leaf node labeled with the highest weight class in D;
6: end if;
7: select a test-attribute with the highest probabilistic information gain ratio to label node N;
8: if (test-attribute is numeric or uncertain numeric or categorical data) then
9: binary split the data from the selected position y;
10: for (each instance R_j) do
11: if (test-attribute $\leq$y) then
12: put it into D_l with weight R_j .w;
13: else if (test-attribute > y) then
14: put it into D_r with weight R_j .w;
15: else
16: put it into D_l with weight R_j .w $*\int_{x1}^{y} f(x)dx$;
17: put it into D_r with weight R_j .w $*\int_{y}^{x2} f(x)dx$;
18: end if;
19: end for;
20: else
21: for (each value a_i(i = 1, . . . , n) of the attribute) do
22: grow a branch D_i for it;
23: end for;
24: for (each instance R_j) do
25: if (test-attribute is uncertain) then
26: put it into Di with R_j .a_i.w*R_j .w weight;
27: else
28: put it into a certain D_i with weight R_j .w;
29: end if
30: end for;
31: end if;
32: for each D_i do
33: attach the node returned by DTU(D_i, att-list);
34: end for;

End

VI. DISS CUSSIONS

a) Uncetainity model

In our discussion, uncertainty models of attributes have been assumed known by some external means. In practice, finding a good model is an application-dependent endeavor.

For example, manufacturers of some measuring instruments do specify in instruction manuals the error of the devices, which can be used as a source of information for modeling error distributions. In some other cases, repeated measurements can be taken and the resulting histogram can be used to approximate the pdf .In the case of random noise, for

example, one could fit a Gaussian distribution5 using the sample mean and variance, thanks to the Central Limit Theorem. During the search for datasets appropriate for our experiments, we have hit a big obstacle: There are few datasets with complete uncertainty information. Although many datasets with numerical attributes have been collected via repeated measurements, very often the raw data has already been processed and replaced by aggregate values, such as the mean. The pdf information is thus not available to us. One example
is the "Breast Cancer" dataset (see Table II) from the UCI repository. This dataset actually contains 10 uncertain numerical features collected over an unspecified number of repeated measurements. However, when the dataset is deposited into the repository, each of these 10 features is replaced by 3 attribute values, giving the mean, the standard score and the mean of the three largest measured values. With these 3 aggregate values, we are unable to recover the distribution of each feature. Even modeling a Gaussian distribution is impossible. These 3 aggregate values are insufficient for us to estimate the variance. Had the people preparing this dataset provided the raw measured values, we would be able to model the pdf's from these values directly, instead of injecting synthetic uncertainty and repeating this for different parameter values for w. Now that we have established in this work that using uncertainty information modeled by pdf's can help us construct more accurate classifiers, it is highly advisable that data collectors preserve and provide complete raw data, instead of a few aggregate values, given that storage is nowadays very affordable.

b) Handling Categorical Attributes

We have been focusing on processing uncertain numerical attributes in this paper. How about uncertain categorical attributes? Like their numerical counterparts, uncertainty can arise in categorical attributes due to ambiguities, data staleness, and repeated measurements. For example, to cluster users based on access logs of HTTP proxy servers using (besides other attributes such as age) the top-level domain names (e.g. ".com", ".edu", ".org", ".jp", ".de", ".ca") as an attribute, we obtain repeated "measurements" of this attribute from the multiple log entries generated by each user. The multiple values collected from these entries form a discrete distribution, which naturally describes the uncertainty embedded in this categorical attribute. The colour of a traffic light signal, which is green at the time of recording, could have changed to yellow or even red in 5 seconds, with probabilities following the programmed pattern of the signal. This is an example of uncertainty arising from data staleness. Colours of flowers recorded in a survey may divide human-visible colours into a number of categories, which may overlap with one another. Such ambiguities could be recorded as a distribution, e.g. 80% yellow and 20% pink. In all these cases, using a distribution to record the possible values (with corresponding probabilities) is a richer representation than merely recording the most likely value.

For a tuple t_i with uncertain categorical attribute A_j, the value uncertainty can be modeled by a discrete probability distribution function $f_{i,j}$:dom$(A_j)\rightarrow$ [0,1] satisfying $\Sigma_{x\epsilon dom(Aj)}$ $f_{i,j}(x) = 1$. This is analogous to the case of uncertain numerical attribute. An internal node n in the decision tree corresponding to a categorical attribute A_j is not associated with a split point, though. Rather, n has many child nodes, each corresponding to a distinct value in dom(A_j). The test to perform at node n is to check the value of A_j, in the test tuple, and the action taken is to follow the branch to the child node corresponding to that attribute value.

To build a decision tree on uncertain data with a combination of numerical and categorical attributes, the same approach as described before can be followed: The tree is built recursively in a top-down manner, starting from the root. At each node, all possible attributes (numerical or categorical) are considered. For each attribute, the entropy of the split is calculated and the attribute giving the highest information gain is selected. The node is assigned that attribute (and split point, if it is a numerical attribute) and the tuples are (fractionally) propagated to the child nodes. Each child node is then processed recursively.

To evaluate the entropy of a categorical attribute Aj, we (fractionally) split the tuples in question into a set of buckets $\{B_v/v\ \epsilon dom(A_j)\}$. Tuple tx is copied into B_v as a new tuple t_y with weight $w_y = f_{x,j}(v)$ if and only if $w_y > 0$. The pdf'sof ty are inherited from t_x, except for attribute A_j, which is set to $f_{y,j}(v) = 1$ and $f_{y,j}(w) = 0$ for all $w\neq v$.The entropy for the split on Aj is calculated using all the buckets. As a heuristic, a categorical attribute that has already been chosen for splitting in an ancestor node of the tree need not be reconsidered, because it will not give any information gain if the tuples in question are split on that categorical attribute again.

VII . Pruning Algorithm

Although UDT can build a more accurate decision tree, it is not as efficient as AVG. As we have explained, to determine the best attribute and split point for a node, UDT has to examine k(ms -1) split points, where k = number of attributes,

c) Pruning by bounding

To handle heterogeneous intervals, we first compute the entropy $H(q,A_j)$ for all end-points $q \in Q_j$. Let H_j^* be the minimum value. Next, for each heterogeneous interval (a, b], we compute a lower bound, Lj, of $H(z,A_j)$ over all $z \in (a, b]$. If $L_j \geq H_j^*$, then no split points $z \in (a, b]$ can give an entropy smaller than $H_j^*$and thus the whole interval can be pruned. This is the basis of our Local Pruning algorithm UDT-LP.A simple but effective improvement on UDT-LP is to use a global (across all attributes A_j) threshold $H^* = \min_{1\leq j\leq k} H_j^*$for pruning. This gives UDT-GP.

d) End-point sampling

We have empirically found that UDT-GP is very effective in pruning intervals, reducing entropy calculation down to only 2.7% of that of UDT. What remains is that UDT-GP spends most time on computing the entropy values of end-points of intervals. As an improvement, we use only 10% of the end-points to derive a pruning threshold. This threshold is slightly less effective, but saves a lot of entropy calculations for the end-points. Using these pruning techniques, we have 2 algorithms.

V . Experiments On Efficiency

The algorithms described above have been implemented in Java using JDK 1.6 and a series of experiments[1] were performed on a PC with an Intel Core 2 Duo 2.66GHz CPU. The data sets used are

Data set	Training tuples	No. of attributes	No. of classes	Test tuples
Japanese vowel	270	12	9	370
Pen Digits	7494	16	10	3498
Page Block	5473	10	5	10-fold
Satellite	4435	36	6	2000
Segment	2310	14	7	10-fold
Vehide	846	18	4	10-fold
Breast Cancer	569	30	2	10-fold
Ionosphere	351	32	2	10-fold
Glass	214	9	6	10-fold
Iris	150	4	3	10-fold

Table II

SELECTED DATA SETS FROM THE UCI MACHINE LEARNING REPOSITORY

a) Pruning Effectiveness

Figure shows that the pruning algorithms are very effective. They reduce the amount of entropy calculations—which dominate the execution time—tremendously, with UDT-ES reaching a >99% reduction ratio.

b) Efficiency

Figure shows that our algorithms are highly efficient, even though they are not as fast as AVG. Please be reminded that AVG generally builds less accurate classifiers. The execution time of AVG is shown in the figure for reference only.

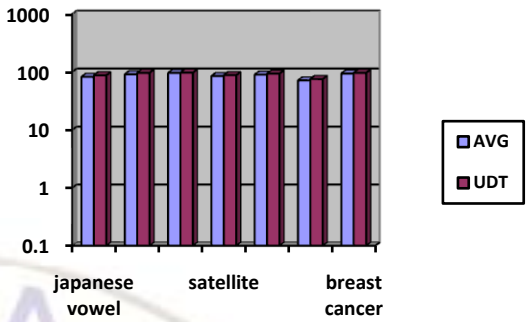


Fig.3. Execution time

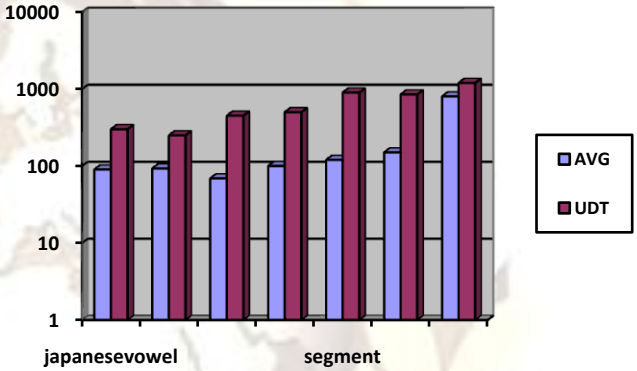


Fig.4.Pruning effectiveness

VI. Conclusion

We have extended the model of decision-tree classification and tree-construction algorithms to accommodate data tuples having numerical attributes with value uncertainty described by arbitrary pdf's. Experiments show that exploiting data uncertainty leads to decision trees with remarkably higher accuracies. Performance is an issue, though, because of the increased amount of information to be processed. We have invented a series of highly effective pruning techniques to improve tree construction efficiency. Pruning by bounding and end-point sampling is novel pruning techniques. Although our novel techniques are primarily designed to handle uncertain data, they are also useful for building decision trees using classical algorithms when there are tremendous amounts of data tuples and also as a future analysis we can handle uncertainty that is caused by categorical data by using these Decision tree algorithms.

VII. References

[1] R. Agrawal, T. Imielinski, and A. N. Swami,"Database mining: A performance perspective," IEEE Trans. Knowl. Data Eng., vol. 5, no. 6, pp. 914–925, 1993.

[2] J. R. Quinlan, "Induction of decision trees," Machine Learning, vol. 1,no. 1, pp. 81–106, 1986.

[3] C4.5: Programs for Machine Learning. Morgan Kaufmann,1993,ISBN 1-55860-238-0.

[4] L.Breiman,J.H. Friedman, R. A. Olshen, andC. J . Stone , Classification and regression Trees. Wadsworth,1984.

[5] Decision Trees for Uncertain Data. Smith Tsang†, Ben Kao†, Kevin Y. Yip‡, Wai- Shing Ho†, Sau Dan Lee†. †Department of Computer Science. 2011.

[6] R.Cheng, D. V. Kalashnikov, and S. Prabhakar,"Querying imprecise data in moving object environments," IEEE Trans. Knowl. Data Eng.,vol. 16, no. 9, pp. 1112– 1127, 2004

[7] W. K. Ngai, B. Kao, C. K. Chui, R. Cheng, M. Chau, and K. Y. Yip, "Efficient clustering of uncertain data," in ICDM. Hong Kong, China:IEEE Computer Society, 18–22 Dec. 2006, pp. 436–445.

[8] S. D. Lee, B. Kao, and R. Cheng, "Reducing UK-means to K-means," in The 1st Workshop on Data Mining of Uncertain Data (DUNE), in conjunction with the 7th IEEE International Conference on Data Mining (ICDM), Omaha, NE, USA, 28 Oct. 2007.

[9] H.-P. Kriegel and M. Pfeifle, "Density-based clustering of uncertain data," in KDD. Chicago,Illinois, USA: ACM, 21–24 Aug. 2005, pp.672–677.

[10] C. K. Chui, B. Kao, and E. Hung, "Mining frequent itemsets from uncertain data," in PAKDD, ser. Lecture Notes in Computer Science, vol. 4426. Nanjing, China: Springer, 22-25 May 2007, pp. 47–58.

[11] C. C. Aggarwal, "On density based transforms for uncertain data mining," in ICDE. Istanbul, Turkey: IEEE, 15-20 Apr. 2007, pp. 866–875.

[12] O. O. Lobo and M. Numao, "Ordered estimation of missing values," in PAKDD, ser. Lecture Notes in Computer Science, vol. 1574. Beijing,China: Springer, 26-28 Apr. 1999, pp. 499–503.

[13] L. Hawarah, A. Simonet, and M. Simonet, "A probabilistic approach to classify incomplete objectsusing decision trees," in DEXA, ser.Lecture Notes in Computer Science, vol. 3180. Zaragoza, Spain: Springer, 30 Aug.-3 Sep. 2004, pp. 549–558.

SUREKHA ALOKAM received her B.Tech in Information & Technology and Engineering from Nalanda Institute Of Engineering and TechnologyCollege, Andhra Pradesh, India, in 2009. She is Pursuing M.Tech in Computer Science and Engineering in JNT University, Kakinada, A.P, India during 2010-2012. Her research invites Data Mining and Knowledge Discovery, Data Warehousing and Database System.

KRISHNAMOHAN ANKALA is working as an ASSOCIATE PROFESSOR under The Department of Computer Science in JNT University, Kakinada, Andhra Pradesh, India.

MHM KRISHNA PRASD is working as an ASSOCIATE PROFESSOR under The Department of Computer Science in JNT University, Vijayanagaram, Andhra Pradesh, India