ISSN: 2278 – 7798
International Journal of Science, Engineering and Technology Research (IJSETR)
Volume 1, Issue 4, October 2012

Generating Decision Trees for Uncertain Data by Using Pruning Techniques

S.Vidya Sagar Appaji,V.Trinadha

Abstract—**Current research techniques on data stream classification mainly focuses on certain data, in which definite and precise value is usually assumed. In this paper, we focus on uncertain data stream classification. Classification is one of the most efficient and uniquely used data mining techniques. In classification, Decision trees can handled with high dimensional data, and their representation is intuitive. Decision trees will handle the data values whose are certain. We extend such classifiers by using decision trees to handle uncertain information. Uncertainty value arises in many applications while the data collection process is performed. For example sources of uncertainty include data staleness and multiple repeated measurements. With uncertainty value, the value of a data item is often represented not only by single value, but also by multiple values forming a probability distribution. Rather than abstracting uncertain data stream by statistical derivatives like mean and median, we extend classical decision tree building algorithms to handle data tuples with uncertain data values.**

Index Terms—**Uncertain Data, Decision Tree, Classification, Data Mining**

I. INTRODUCTION

In this modern world, huge amount of information is kept in the databases. Thus data-mining can be very effective for extracting knowledge from huge amount of data. Classification has many applications in real world, such as stock planning of large superstores, medical diagnosis, etc. Classification is separation or ordering of objects into classes. There are various classification techniques i.e. Decision tree K-nearest neighbor, Naïve bays classifier, neural network. We identify the following characters of uncertain data stream classification:

A. Data with uncertainty: uncertain values have to be handled cautiously, or else the mining results could be unreliable or even wrong

B. High speed input data: the classifier should have the ability to process huge volume of data which arrive at high speed.

C. Concept-drift detecting: the classifier should be capable of detecting and responding to changes in the example-generating process.

D. Limited memory space: only limited memory space is available to the classifier, which means that the classifier has to scan the input samples for only once.

In this paper we discuss decision tree classification. Classification is a classical problem in machine learning and data mining. It summarizes an approach for synthesizing decision trees that has been used in a variety of systems, and it describes one such system, ID3, in detail. Decision trees are mainly used for handling "Decision-making". Many algorithms, such as ID3 and C4.5, have been devised for decision tree construction. C4.5 is an extension to ID3 algorithm. In traditional decision-tree classification, a feature of a tuple is either categorical or numerical. For the latter, a precise and definite point value is usually assumed. In many applications, however, data uncertainty is common. The value of a feature/attribute is thus best captured not by a single point value, but by a range of values giving rise to a probability distribution. A simple way to handle data uncertainty is to abstract probability distributions by summary statistics such as means and variances. We call this approach "Averaging". Another approach is to consider the complete information carried by the probability distributions to build a decision tree. We call this approach "Distribution based". Our goals are to devise an algorithm or building decision trees from uncertain data using the Distribution based approach; to investigate whether the Distribution based approach could lead to a higher classification accuracy compared with the Averaging approach;

and to establish a theoretical foundation on which pruning techniques are derived that can significantly improve the computational efficiency of the Distribution-based algorithms.

II. RELATED WORK

Decision trees are one of the most important aspects for "Decision-making". Classification is one of the most widespread data mining problems found in real life. There has been a growing interest in uncertain data mining. The well-known k-means clustering algorithm has been extended to the UK-means algorithm and various pruning techniques have been proposed, Decision tree classification with missing data has been addressed for decades in the form of missing values. In C4.5, these are handled by using *fractional tuples*. In this work, we adopt the technique of fractional tuple for splitting tuples into subsets when the domain of its pdf spans across the split point. However, handling data values represented as pdf's is unprecedented. In fuzzy decision tree classification, both attributes and class labels can be fuzzy and are represented in fuzzy terms. Given a fuzzy attribute of a data tuple, a degree is assigned to each possible value, showing the extent to which the data tuple belongs to a particular value. Our work instead gives classification results as a distribution: for each test tuple, we give a distribution telling how likely it belongs to each class. Building a decision tree on tuples with numerical, point valued data is computationally demanding. Finding the best split point is computationally expensive. To improve efficiency, many techniques have been proposed to reduce the number of candidate split points. Our work can be considered an extension of these optimization techniques.

A. Uncertain Data Classification

Numerous uncertain data classification algorithms have been proposed in the literature in recent years. Qin et al. (2009) proposed a rule-based classification algorithm for uncertain data. Rental. (2009) proposed to apply Naive Bays approach to uncertain data classification problem. Decision tree model is one of the most popular classification models because of its advantages (Tsang et al. 2009, Quinlan 1993). Several decision tree based classifiers for uncertain data are proposed by research community. The well-known C4.5 classification algorithm was extended to the DTU (Qin et al. 2009a) and the UDT (Tsang et al. 2009) for classifying uncertain data. (Qin et al. 2009a) used probability vector and probability density function (pdf) to represent uncertain categorical attribute (Qin et al. 2009b) and uncertain numerical attribute (Cheng et al. 2003) respectively. They constructed a well performance decision tree for uncertain data (DTU). Tsang et al. (2009) used the "complete information" of pdf to construct a uncertain decision tree(UDT) and proposed a series of pruning techniques to improve the efficiency. Both DTU and UDT algorithm are extensions of C4.5, they can only be used to deal with uncertain static data set, while our UCVFDT algorithm is capable of handling uncertain data stream, in which huge volume of data arrive at high speed. 210 Decision Tree for Dynamic and Uncertain Data Streams

B. Data Stream Classification

There are two main approaches for classifying data streams: single classifier based approach and ensemble based approach. For single classifier based approach, the most well-known classifiers are VFDT and CVFDT. The CVFDT improves the VFDT with ability to deal with concept drift. After that, many decision tree based algorithms were proposed for data stream classification (for example, Gametal. 2005, Gametal. 2006). For ensemble based approaches, the initial papers use static majority voting (for example, Street and Kim 2001, Wang et al. 2003), while the current trend is to use dynamic classifier ensembles (for example, Zhang and Jin 2006, Zhu et al. 2004). All of algorithms mentioned above can only handle certain data. Pan et al. (2009) proposed two types of ensemble based algorithms, Static Classifier Ensemble (SCE) and Dynamic Classifier Ensemble (DCE) for mining uncertain data streams. To the best of our knowledge, this is the only work devoted to classification of uncertain data streams. However, in (Pan et al. 2009), class value of a sample is assumed to be uncertain, while attributes is assumed to have precise value. Our UCVFDT handles uncertain attributes, while class value is assumed to be precise.

C. Proposed System

C4.5 belongs to a succession of decision tree learners that trace their origins back to the work of Hunt and others in the late 1950s and early 1960s (Hunt 1962). Its immediate predecessors were ID3 (Quinlan 1979), a simple system consisting initially of about 600 lines of Pascal, and C4 (Quinlan 1987). C4.5 has grown to about 9,000 lines of C that is available on diskette with Quinlan (1993). Although C4.5 has been superseded by C5.0, a commercial system from Rule Quest Research, this discussion will focus on C4.5 since its source code is readily available.

This section explains one of the algorithms used to create Univariate DT's. This one, called C4.5, is based on the ID32 algorithm, that tries to find small (or simple) DT's. We start presenting some premises on witch this algorithm is based, and after we discuss the inference of the weights and tests in the nodes of the trees.

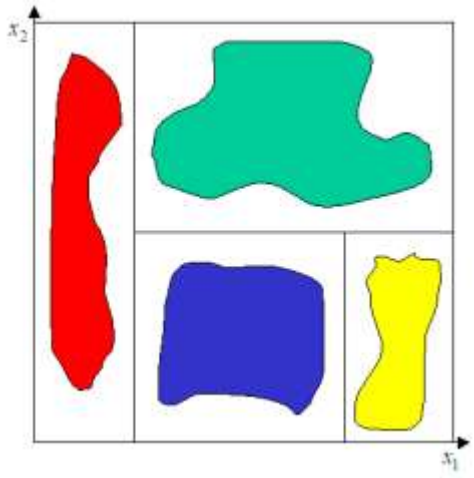

Fig.1 Partitions created in a DT.

D. Construction

Some premises guide this algorithm, such as the following

• If all cases are of the same class, the tree is a leaf and so the leaf is returned labeled with this class;

• For each attribute, calculate the potential information provided by a test on the attribute (based on the probabilities of each case having a particular value for the attribute). Also calculate the gain in information that would result from a test on the attribute (based on the probabilities of each case with a particular value for the attribute being of a particular class);

• Depending on the current selection criterion, find the best attribute to branch on.

Decision trees are one of the most important aspect for "Decision-making". Classification is one of the most widespread data mining problems found in real life. Decision tree classification is one of the best-known solution approaches. ID3, first proposed by Quinlan is a particularly elegant and instinctive solution [4]. This article presents an algorithm for privately building an ID3 decision tree. While this has been done for horizontally partitioned data [5], Liddell *et al* has proposed a secure algorithm to build a decision tree using ID3. Data uncertainty has been broadly classified as existential uncertainty and value uncertainty. There has been a growing interest in uncertain data mining. In [6], the well-known k-means clustering algorithm is extended to the UK-means algorithm [7][8] for clustering uncertain data. Data uncertainty is usually captured by pdf's, which are generally represented by sets of sample values. Mining uncertain data is therefore computationally costly due to information explosion (sets of samples vs. single values). To improve the performance of UK-means, pruning techniques have been proposed. Examples include min-max dist pruning [9] and CK-means [10]. In C4.5 and probabilistic decision trees[1], missing values in training data are handled by using fractional tuples. During testing, each missing value is replaced by multiple values with probabilities based on the training tuples, thus allowing probabilistic classification results. In this work, we adopt the technique of fractional tuple for splitting tuples into subsets when the domain of its pdf spans across the split point. To improve efficiency, many techniques have been proposed to reduce the number of candidate split points. We use density-based clustering (e.g., FDBSCAN), frequent item set mining , and density-based classification.

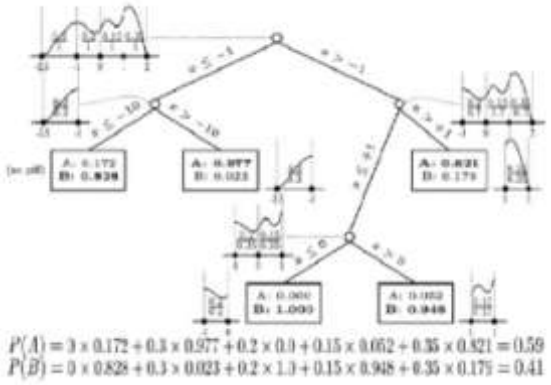

Fig.2 classifying a test tuple

III. APPROACHES

There are two approaches for handling uncertain data. The first approach, called "Averaging," transforms an uncertain data set into a point-valued one by replacing each pdf with its mean value. More specifically , the mean value $v_{i,j} = \int_{a_{i,j}}^{b_{i,j}} x\, f_{i,j}(x)\, dx$ as its representative value .The feature vector of ti is thus transformed to ($v_{i,1},\ldots\ldots\ldots,v_{i,k}$) A decision tree can then be built by applying a traditional tree construction algorithm. Several identical algorithms have been introduced for decision tree construction. This work provides the ID3 classification algorithm. Very simply, ID3 builds a decision tree from a fixed set of examples. The resulting tree is used to classify future samples. **C4.5** is an algorithm used to generate a decision tree developed by Ross Quinlan. C4.5 is an extension of Quinlan's earlier ID3 algorithm. The decision trees generated by C4.5 can be used for classification, and for this reason, C4.5 is often referred to as a statistical classifier. We present details of the tree construction algorithms under the two approaches in the following sections.

TABLE 1
Example Tuples

tup.c	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	

To exploit the full information carried by the pdfs, our second approach, called "Distribution-based," considers all the sample points that constitute each pdf. The challenge here is that a training tuple can now "pass" a test at a tree node probabilistically when its pdf properly contains the split point of the test.

A. Averaging

A straightforward way to deal with the uncertain information is to replace each pdf with its expected value, thus, effectively converting the data tuples into point-valued tuples. This reduces the problem back to that for point valued data, and hence, traditional decision tree algorithms such as ID3 and C4.5 can be reused. We call this approach Averaging (AVG).

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 $P_n(c) = 1, P_n(C^1) = 0$ $\forall c^1 \neq C$. 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 $v_{i,jn} <= z_n$ are put in the "left" subset L; the rest goes to the "right" subset R. If either L or R is empty (even after exhausting all possible choices of A and z), 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 Pn. In particular, for each class label $c \in C$, we assign to Pn(c) the fraction of tuples in S that is labeled 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.

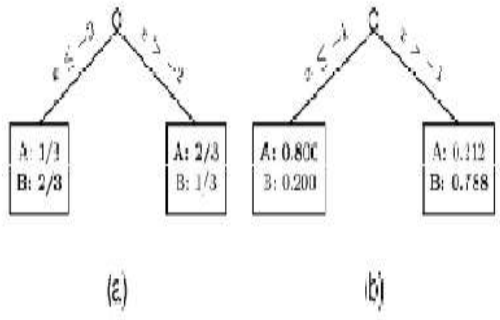

Fig.3 Decision tree built from example tuples in table1.(a)Averaging (b)Distribution based

Let us illustrate this classification algorithm using the example tuples shown in Table 1. This set consists of six tuples of two class labels "A" and "B." Each tuple has only one attribute, whose (discrete) probability distribution is shown under the column "probability distribution." For instance, tuple 3 has class label "A" and its attribute takes the values of -1, +1, +10 with probabilities 5/8, 1/8, 2/8, respectively. The column "mean" shows the expected value of the attribute. For example, tuple 3 has an expected value of +2.0. With Averaging, there is only one way to partition the set: the even-numbered tuples go to L and the odd-numbered tuples go to R. The tuples in each subset have the same mean attribute value, and hence, cannot be discerned further. The resulting decision tree is shown in Fig. 2a. Since the left subset has 2 tuples of class B and 1 tuple of class A, the left leaf node L has the probability distribution P_L (A) =1/3 and P_L(B) =2/3 over the class labels. The probability distribution of class labels in the right leaf node R is determined analogously. Now, if we use the six tuples in Table 1 as test tuples4 and use this decision tree to classify them, we would classify tuples 2,4,6 as class "B" (the most likely class label in L), and hence, misclassify tuple 2. We would classify tuples 1, 3, 5 as class "A," thus getting the class label of 5 wrong. The accuracy is 2=3.

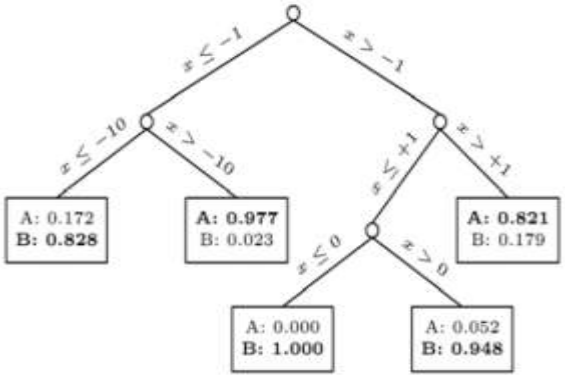

Fig.4 Example decision tree before post pruning

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. We assume that entropy is used as the measure since it is predominantly used for building decision trees. For each of the (m-1)k combinations of attributes (A_j)and split points (z), we divide the set S into the "left" and "right" subsets L and R. We then compute the entropy for each such
Combination:

$$H(z,A_j)=\sum_{X=L,R}\frac{|X|}{|S|}\left(\sum_{c\in C}-P_{C/X}\log_2 P_{C/X}\right)$$

Where P_c=X is the fraction of tuples in X that is labeled c.

B. Distribution-Based Approach

For 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 have 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 $t_j \in S$ under attribute A_{jn} spans the interval $[a_{i,jn},b_{i,jn}]$. If $b_{i,jn} <= z_n$, the pdf of ti lies completely on the left of the split point, and thus, t_j is assigned to L. Similarly, we assign t_i to R if$z_n.<a_{i,jn}$ If the pdf properly contains the split point, i.e., $a_{i,jn} <= z_n < b_{i,jn}$, we split ti into two fractional tuples t_L and add them to L and R, respectively. We call this algorithm Uncertain Decision Tree (UDT). Let us reexamine the example tuples in Table 1 to see how the distribution-based algorithm can improve classification accuracy. By taking into account the probability distribution, UDT

builds the tree shown in Fig. 3 before pre pruning and post pruning are applied. This tree is much more elaborate than the tree shown in Fig. 2a because we are using more information, and hence, there are more choices of split points. The tree in Fig. 3 turns out to have a 100 percent classification accuracy. After post pruning, we get the tree in Fig. 2b. Now, let us use the six tuples in Table 1 as testing tuples to test the tree in Fig. 2b. 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 six tuples are classified correctly. This handcrafted example thus illustrates that by considering probability distributions rather than just expected values, we can potentially build a more accurate decision tree.

Data Set	Training Tuples	No. of Attributes	No. of Classes	Test Tuples
JapaneseVowel	270	12	9	370
PenDigits	7494	16	10	3498
PageBlock	5473	10	5	10-fold
Satellite	4435	36	6	2000
Segment	2310	14	7	10-fold
Vehicle	846	18	4	10-fold
BreastCancer	569	30	2	10-fold
Ionosphere	351	32	2	10-fold
Glass	214	9	6	10-fold
Iris	150	4	3	10-fold

TABLE 2 Selected Data Sets from the UCI Machine Learning Repository

IV. THE C5.0 CLASSIFIER

The C5.0 algorithm is a new generation of Machine Learning Algorithms (MLAs) based on decision trees. It means that the decision trees are built from list of possible attributes and set of training cases, and then the trees can be used to classify subsequent sets of test cases. C5.0 was developed as an improved version of well-known and widely used C4.5 classifier and it has several important advantages over its ancestor. The generated rules are more accurate and the time used to generate them is lower (even around 360 times on some data sets). In C5.0 several new techniques were introduced:

- Boosting: several decision trees are generated and combined to improve the predictions.
- Variable misclassification costs: it makes it possible to avoid errors which can result in harm.
- New attributes: dates, times, timestamps, ordered discrete attributes.
- Values can be marked as missing or not applicable for particular cases.
- Supports sampling and cross-validation.

The C5.0 classifier contains a simple command-line interface, which was used by us to generate the decision trees, rules and finally test the classifier. In addition a free C source code for including C5.0 classifier in external applications is available on the C5.0 website. Detailed description of C5.0 and all its options and abilities is published in the tutorial.

V. EXPERIMENTS ON ACCURACY

In order to achieve a higher classification accuracy by considering data uncertainty, we should implement AVG and UDT and apply them to 10 real data sets (see Table 2) taken from the UCI Machine Learning Repository [17]. These data sets are chosen because they mostly contain numerical attributes obtained from the measurements. Here, first data set contains 640 tuples, each representing an utterance of the Japanese vowels by one of the nine participating speakers. Each tuple contains 12 numerical attributes, which are Linear Predictive Coding (LPC) coefficients. These coefficients reflect important features of speech sound. Each attribute value consists of 7-29 samples of LPC coefficients collected over time. These samples represent uncertain information and are used to model the pdf of the attribute for the tuple. The other nine data sets contain "point values" without uncertainty.

Data Set	AVG	UDT Best Case	Gaussian Distribution w=1%	Gaussian Distribution w=5%	Gaussian Distribution w=10%	Gaussian Distribution w=20%	Uniform Distribution w=2%	Uniform Distribution w=10%	Uniform Distribution w=20%
JapaneseVowel	81.89	87.30	*87.30 (The distribution is based on samples from raw data)						
Pen-Digit	90.87	96.11	91.56	92.18	93.79	95.22	91.68	93.76	*96.11
PageBlock	95.73	96.82	*96.82	96.32	95.74	94.87#	N/A		
Satellite	84.48	87.73	85.18	87.1	*87.73	86.25	85.9	87.2	85.9
Segment	89.37	92.91	91.91	*92.91	92.23	89.11#	N/A		
Vehicle	71.03	75.09	72.44	72.98	73.18	*75.09	69.97#	71.04	71.62
BreastCancer	93.52	95.93	94.73	94.28	95.51	*95.93	N/A		
Ionosphere	88.69	91.69	89.65	88.92	*91.69	91.6	N/A		
Glass	66.49	72.75	69.6	*72.75	70.79	69.69	N/A		
Iris	94.73	96.13	94.47#	95.27	96	*96.13	N/A		

TABLE 3Accuracy Improvement by Considering the Distribution

We enhance uncertainty information by fitting appropriate error models on to the point data. For each tuple t_i and for each attribute Aj, the point value $v_{i,j}$ reported in a data set is used as the mean of a pdf $f_{i,j}$, defined over an interval $[a_{i,j}; b_{i,j}]$. The range of values for A_j (over the whole data set) is noted and the width of $[a_{i,j}, bi_{,j}]$ is set to w. $|A_j|$, where $|A_j|$ denotes the width of the range for A and w is a controlled parameter. The results of applying AVG and UDT to the 10
data sets are shown in Table 3. Using C4.5 [3] the information gain is calculated. In the experiments, each pdf is represented by 100 sample points (i.e., s=100), except for the Japanese Vowel" data set. We have repeated the experiments using various values for w. For most of the data sets, Gaussian distribution is assumed as the error model. Since the data sets "Pen Digits," "Vehicle," and "Satellite" have integer domains, we suspected that they are highly influenced by quantization noise. So, we have also tried uniform distribution on these three data sets, in addition to Gaussian.6For the "Japanese Vowel" data set, we use the uncertainty given by the raw data (7-29 samples) to model the pdf.

VI. EFFECT OF NOISE MODEL

In the experiment above, we have taken data from the UCI repository and directly added uncertainty to it so as to test our UDT algorithm. The amount of errors in the data is uncontrolled. So, in the next experiment, we inject some artificial noise into the data in a controlled way except the "Japanese Vowel". For each tuple t_i and for each attribute A_j, the point value $v_{i,j}$ is perturbed by adding a Gaussian noise with zero mean and a standard deviation equal to ~=1/4(u. |A|), where u is a controllable parameter. So, the perturbed value is $v_{i,j}= v_{i,j}+\Delta_{i,j}$ where $\Delta_{i,j}$ is a random number which Follows $N(0,\sigma^2)$.

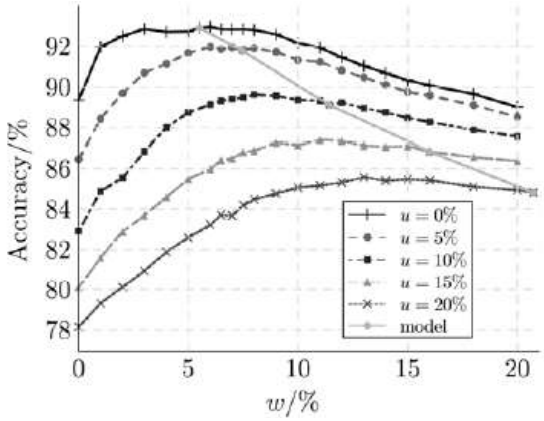

Fig.5 Experiment with controlled noise on data set "segment"

Each curve in the figure corresponds to one value of u, which controls the amount of perturbation that has been artificially added to the UCI data set. The x-axis corresponds to different values of w—the error model that we use as the uncertainty information. The y-axis gives the accuracy of the decision tree built by UDT.

VII. PRUNING ALGORITHMS

Pruning Empty and Homogeneous Intervals: Definition 1: For a given set of tuples S, an optimal split point for an attribute A_j is one that minimizes $H(z,A_j)$. (Note that the minimization is taken over all z €[q1, qv].).The end points define v _ 1 disjoint intervals: (qi, qi+1] for i =1; . . . ; v -1. We will examine each interval separately. For convenience, an interval is denoted by (a,b].

Definition 2 (Empty interval): An interval (a,b] is empty if

$$\int_a^b f_{h,j}(x)dx = 0 \quad \text{for all } t_h \in S.$$

Definition 3 (Homogeneous interval): An interval (a, b] is homogeneous if there exists a class label c ∈ C such that

$$\int_a^b f_{h,j}(x)dx \neq 0 \rightarrow c_h = c \text{ for all } t_h \in S.$$

Definition 4 (Heterogeneous interval): An interval (a,b] is heterogeneous if it is neither empty nor homogeneous.

Theorem 1: If an optimal split point falls in an empty interval, then an end point of the interval is also an optimal split point.

Proof: By the definition of information gain, if the optimal split point can be found in the interior of an empty interval (a,b], then that split point can be replaced by the end point a without changing the resulting entropy. As a result of this theorem, if (a,b] is empty, we only need to examine the end point a when looking for an optimal split point. There is a well-known analogue for the point data case, which states that if an optimal split point is to be placed between two Consecutive attribute values, it can be placed anywhere in the interior of the interval and the entropy will be the same [28]. Therefore, when searching for the optimal split point, there is no need to examine the interior of empty intervals. The following theorem further reduces the search space:

Theorem 2: If an optimal split point falls in a homogeneous interval, then an end point of the interval is also an optimal split point.

Definition 5 (Tuple density): Given a class c $\in$ C, an attribute A_j, and a set of tuples S, we define the tuple density function $g_{c,i}$ as

$$g_{c,j} = \sum_{t_h \in S : c_h = c} w_h f_{h,j}$$

where w_h is weight of the fractional tuple $t_h \in S$

Definition 6 (Tuple count): For an attribute A_j, the tuple count for class c $\in$ C in an interval (a, b] is

$$\gamma_{c,j}(a,b) = \int_a^b g_{c,j}(x)dx$$

Theorem 3: Suppose that the tuple count for each class increases linearly in a heterogeneous interval (a, b] (i.e.,

$$\forall_c \in C, \forall t \in [0,1], \gamma_{c,j}(a,(1-t)a+tb) = \beta_c t$$

for some constant β_c). If an optimal split point falls in ða; b_, then an end point of the interval is also an optimal split point.

VIII. END POINT SAMPLING

UDT-GP Global Pruning algorithm is very effective in pruning intervals. In some settings, UDT-GP reduces the number of "entropy calculations" (including the calculation of entropy values of the split points and the calculation of entropy-like lower bounds for intervals) to only 2.7 percent of that of UDT.

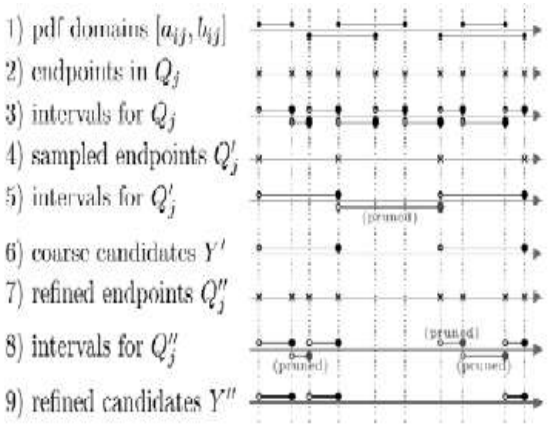

Fig.6 Illustration of end point sampling.

In fig, Row 1 shows the intervals obtained from the domains of the pdfs. The collection of end points of these intervals constitutes the set Q_j(row 2). From these end points, disjoint intervals are derived (row 3). Instead of using the set of all end points Q_j(row 2), we take a sample Q_{j1}(row 4) of these points. The algorithm thus operates on the intervals derived from Q_{j1} (row 5) instead of those derived from Q_j(row 3). After all the prunings on the coarser intervals are done, we are left with a set Y 0 of candidate intervals (row 6). (Note that a couple of end points are pruned in the second interval of row 5.) For each un pruned candidate interval q_y ,q_{y+1} in row 6, we bring back the original set of end points inside the interval (row 7) and their original finer intervals (row 8). The candidate set of intervals obtained after pruning is Y^{11} (row 9), which is a much smaller candidate than the set of candidate intervals when no end point sampling is used. We incorporate these end point sampling strategies into UDT-GP. The resulting algorithm is called UDT-ES.

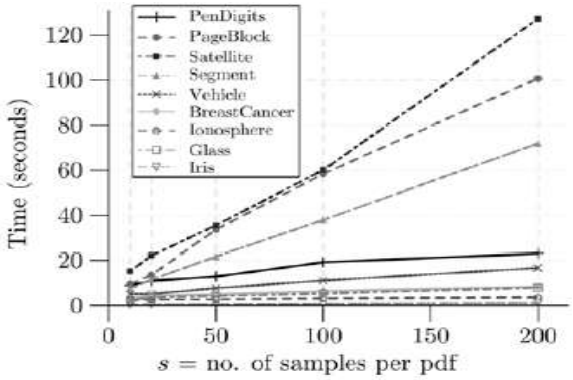

Fig.7 Effects of s on UDT-ES.

IX. CONCLUSION

In this work, we have seen the uncertain data is handled through "Averaging" by using means and variances. But in "Distribution-based" the accuracy of a uncertain data is detected through decision trees. Decision trees calculate the entropy measure and enhance the information gain for better accuracy. Several procedures and algorithm handles data uncertainty. We exploit data uncertainty that leads to decision trees with remarkably higher accuracies.

ACKNOWLEDGMENT

It gives me immense pleasure to express deep sense of gratitude to my guide **Mr. S. Vidya Sagar Appaji,** Assistant Professor, Department of Computer Science & Engineering, for his Whole-hearted and invaluable guidance throughout the work. Without his sustained and sincere effort, this work would not have taken his shape. He encouraged and helped me to overcome various difficulties that I have faced at various stages of paper.

REFERENCES

[1] J. R. Quinlan, "Induction of decision trees," Machine Learning, vol. 1, no. 1, pp. 81–106, 1986.J. R. Quinlan, "Learning logical definitions from relations," Machine Learning, vol. 5, pp. 239–266, 1990.

[2] J. Kinoshita, "Fuzzy decision trees by fuzzy ID3 algorithm and its application to diagnosis systems," in Fuzzy Systems, 1994, vol. 3. IEEE World Congress on Computational Intelligence, 26-29 Jun. 1994, pp.2113–2118.

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

[4] J.R. Quinlan, "Induction of decision trees," In Jude W.Shavlik, Thomas G. Dietterich, (Eds.), Readings in Machine Learning. Morgan Kaufmann, 1990. Originally published in Machine Learning, vol. 1, 1986, pp 81–106.

[5] Y. Lindell, B. Pinkas, "Privacy preserving data mining," In Journal of Cryptology vol. 15, no. 3, 2002, pp 177–206.

[6] M. Chau, R. Cheng, B. Kao, and J. Ng, "Uncertain data mining: An example in clustering location data," in PAKDD, ser. Lecture Notes in Computer Science, vol. 3918. Singapore: Springer, 9–12 Apr. 2006, pp. 199–204

[7]J. Chen and R. Cheng, "Efficient Evaluation of Imprecise Location- Dependent Queries," Proc. Int'l Conf. Data Eng. (ICDE), pp. 586- 595, Apr. 2007.

[8] M. Chau, R. Cheng, B. Kao, and J. Ng, "Uncertain Data Mining: An Example in Clustering Location Data," Proc. Pacific-Asia Conf. Knowledge Discovery and Data Mining (PAKDD), pp. 199-204, Apr. 2006.

[9] 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.

[10] 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.

[11] T. Elomaa and J. Rousu, "General and Efficient Multi splitting of Numerical Attributes," Machine Learning, vol. 36, no. 3, pp. 201- 244, 1999.

[12] U.M. Fayyad and K.B. Irani, "On the Handling of Continuous- Valued Attributes in Decision Tree Generation," Machine Learning, vol. 8, pp. 87-102, 1992.

[13] T. Elomaa and J. Rousu, "Efficient Multi-splitting Revisited: Optima- Preserving Elimination of Partition Candidates," Data Mining and Knowledge Discovery, vol. 8, no. 2, pp. 97 126, 2004.

[14]H.-P. Kriegel and M. Pfeifle, "Density-Based Clustering of Uncertain Data," Proc. Int'l Conf. Knowledge Discovery and Data Mining (KDD), pp. 672-677, Aug. 2005.

[15] C.K. Chui, B. Kao, and E. Hung, "Mining Frequent Item sets from Uncertain Data," Proc. Pacific-Asia Conf. Knowledge Discovery and Data Mining (PAKDD), pp. 47-58, May 2007.

[16] C.C. Aggarwal, "On Density Based Transforms for Uncertain Data Mining," Proc. Int'l Conf. Data Eng. (ICDE), pp. 866-875, Apr. 2007.

[17] A. Asuncion and D. Newman, UCI Machine Learning Repository, http://www.ics.uci.edu/mlearn/MLRepository.html, 2007.

S.VIDYA SAGAR APPAJI received the M.Tech degree from JNTU Kakinada. He has five years of teaching experience. He is currently working as Assistant Professor in M.V.G.R College of engineering .He has three papers in journals

V.TRINADHA received B.Tech degree from G.M.R.I.T .Currently pursuing M.Tech (CSE) from JNTU Kakinada University.